Redundancy in Web-scale Information Extraction: Probabilistic Model and Experimental Results

Douglas C. Downey

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Washington

2008

Program Authorized to Offer Degree: Computer Science and Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Douglas C. Downey

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of the Supervisory Committee:

____
Oren Etzioni

Reading Committee:

____
Pedro Domingos

____
Oren Etzioni

____
Eric Horvitz

Date: ______________________

University of Washington

Abstract

Redundancy in Web-scale Information Extraction: Probabilistic Model and Experimental Results

Douglas C. Downey

Chair of the Supervisory Committee:
Professor Oren Etzioni
Computer Science & Engineering

Information Extraction (IE) is the task of automatically extracting knowledge from text. The massive body of text now available on the World Wide Web presents an unprecedented opportunity for IE. IE systems promise to encode vast quantities of Web content into machine-processable knowledge bases, presenting a new approach to a fundamental challenge for artificial intelligence: the automatic acquisition of massive bodies of knowledge. Such knowledge bases would dramatically extend the capabilities of Web applications. Future Web search engines, for example, could query the knowledge bases to answer complicated questions that require synthesizing information across multiple Web pages.

However, IE on the Web is challenging due to the enormous variety of distinct concepts expressed. All extraction techniques make errors, and the standard error-detection strategy used in previous, small-corpus extraction systems—hand-labeling examples of each concept to be extracted, then training a classifier using the labeled examples—is intractable on the Web. How can we automatically identify correct extractions for arbitrary target concepts, *without* hand-labeled examples?

This thesis shows how IE on the Web is made possible through the *KnowItAll hypothesis*, which states that extractions that occur more frequently in distinct sentences in a corpus are more likely to be correct. The KnowItAll hypothesis holds on the Web, and can be used to identify many correct extractions because the Web is highly *redundant*: individual

facts are often repeated many times, and in many different ways. In this thesis, we show that a probabilistic model of the KnowItAll hypothesis, coupled with the redundancy of the Web, can power effective IE for arbitrary target concepts *without* hand-labeled data. In experiments with IE on the Web, we show that the probabilities produced by our model are 15 times better, on average, when compared with techniques from previous work. We also prove formally that under the assumptions of the model, “Probably Approximately Correct” IE can be attained from only unlabeled data.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Number Page

LIST OF TABLES

ACKNOWLEDGMENTS

This work was made possible through the assistance of a number of people. I will start by thanking my student collaborators from the University of Washington. Stef Schoenmackers is the consummate team player, extremely smart and a genuine pleasure to work with; the Rel-grams model and the experiments in Chapter 4 of this dissertation include a number of his contributions. I'd like to thank everyone in the KnowItAll Project group at UW—especially Mike Cafarella, Alan Ritter, Alex Yates, Matthew Broadhead, and Michele Banko—for providing helpful discussions and productive collaborations. Mike and Michele deserve special thanks for devising and implementing the TextRunner architecture, an indispensable research platform for a number of students including myself.

I'd also like to thank the UW Computer Science and Engineering faculty whose teaching and mentoring were critical to my progress. Pedro Domingos served on my thesis committee, and his advice (and his excellent machine learning courses) made a large positive impact on my work. Pedro took the time to ask great questions and offer key suggestions for my research throughout—from pointing me toward Good-Turing smoothing techniques for the original Urns paper, to suggesting important baselines for comparison for the final experiments in the thesis. I also greatly enjoyed the chances I had to work with Dan Weld. Dan's incisive questions deepened my understanding of my work, and his suggestions for research directions and presentation strategies were always helpful.

I'd like to thank Stephen Soderland, who was instrumental in pushing the original KnowItAll system to completion. Many of us benefited greatly from using Stephen's work as a foundation for our research. I appreciate Stephen's research advice throughout my graduate school career, and particularly his contributions to the Urns paper (Chapter 3 of this dissertation).

I was lucky enough to intern at Microsoft Research during graduate school; the people and the research resources available there made it an incredibly valuable experience. I'd like to thank my mentor at MSR, Susan Dumais. Sue's encyclopedic knowledge of Information Retrieval made it possible for me to work on important research questions right away, despite my lack of expertise in that area initially. I also appreciate Sue's willingness to drill down into the finest experimental details and clarify key methodological points, which strengthened our results.

I'd like to thank my manager at MSR, Eric Horvitz, who also served on my thesis committee. Eric produced a tremendous number of ideas that went into the three papers that he, Sue, Dan Liebling and myself collaborated on during and after my internship at Microsoft Research. In addition, Eric offered guidance and advice on my thesis research, and I greatly appreciated having his unique and creative perspective on my committee.

I'd like to thank my advisor at UW, Oren Etzioni. Oren put me in an ideal position to succeed in graduate school by suggesting fruitful research directions, bringing together great student collaborators and resources, and most importantly teaching me how to ask and answer research questions. Working with Oren on tough research challenges was both exhilarating and demanding—exactly what graduate school should be.

What I appreciated most about Oren was that he made it a top priority to help his graduate students reach their potential. He always had the time to offer me the guidance I needed—as an advisee, you wanted to work hard for Oren because he was working hard for you. I can't say enough about how rewarding and intellectually exciting it was to have Oren as an advisor.

I was aided at various times and in various important ways by a number of others, including: Randy Beer, Dave Preston, Julie and Jeremy Fleischer, Hillel Chiel, George Voutsadakis, Emil Biga, Tom Bogart, Lindsay Michimoto, Raj Rao, Patrick Allen, and everyone in the 590bt seminar.

Lastly, I'd like to thank my wife Lisa, to whom this thesis is dedicated; her tireless support made it possible. And I'd like to thank my parents—in multiple subjects, one of which was mathematics, they were the best teachers I ever had.

DEDICATION

To Lisa

Chapter 1
INTRODUCTION

Web search engines are invaluable tools for answering questions: a user curious about the identity of the "1946 World Series champion," for example, need only type that phrase into a search engine to immediately find a page that mentions the answer. However, despite their utility, search engines can address only a fraction of the questions users attempt to answer on the Web. Consider, for example, a job seeker in search of a list of nanotechnology companies hiring on the West Coast, or a computer buyer needing a summary of reviews on the Web for a particular laptop. The answers to the questions can be found on the Web, but not on just a single page. Obtaining the answers requires extracting and synthesizing information from multiple documents—using existing Web search engines, this is a tedious and error-prone manual process.

Recently, systems such as KnowItAll [23, 24, 26], TextRunner [5, 6], and others [45] have attempted to automate the collection of facts from the Web. Automatically extracting knowledge from text is the task of *Information Extraction* (IE). Applied to the Web, IE promises to radically improve Web search engines, allowing them to answer complicated questions by synthesizing information across multiple Web pages. Further, extraction from the Web presents a new approach to a fundamental challenge in artificial intelligence: the automatic accumulation of massive bodies of knowledge.

IE on the Web is particularly challenging due to the variety of different concepts expressed. The strategy employed for previous, small-corpus IE is to hand-label examples for each target concept, and use the examples to train an extractor [28, 56, 9, 13, 42, 38]. On the Web, hand-labeling examples of each concept is intractable—the number of concepts of interest is simply far too large. IE *without* hand-labeled examples is referred to as *Unsupervised* Information Extraction (UIE). UIE systems such as KnowItAll, TextRunner, and others have demonstrated that at Web scale, automatically-generated textual patterns can perform UIE for millions of diverse facts. As a simple example, an occurrence of the phrase "C such as x" suggests that the string x is a member of the class C, as in the phrase "films such as Star Wars."

However, all extraction techniques make errors, and a key problem for any IE system is determining which of its extractions are correct. In UIE, where labeled examples are unavailable, the task is particularly challenging. How can we automatically identify correct extractions for arbitrary target concepts, *without* hand-labeled examples?

This thesis introduces a solution to the above question that applies across a broad spectrum of UIE systems and techniques. It relies on the *KnowItAll hypothesis*, which states that extractions that occur more frequently in distinct sentences in a corpus are more likely to be correct. The KnowItAll hypothesis holds on the Web, and serves as a Web-scale complement to the fundamental *distributional hypothesis* of language, which states that words with similar meanings tend to appear in similar contexts.

KnowItAll hypothesis: Extractions drawn more frequently from distinct sentences in a corpus are more likely to be correct.

Intuitively, we would expect the KnowItAll hypothesis to hold because although extraction errors occur (e.g., KnowItAll erroneously extracts `California` as a `City` name from the phrase "states containing large cities, such as California"), errors occurring in distinct sentences tend to be different. Thus, typically a given erroneous extraction is repeated only a limited number of times. Further, while the Web does contain some misinformation (for example, the statement "Elvis killed JFK" appears almost 200 times on the Web according to a major search engine), this tends to be the exception (the correct statement "Oswald killed JFK" occurs over 3000 times).

At Web-scale, the KnowItAll hypothesis can identify many correct extractions due to *redundancy*: in large corpora like the Web, individual facts are often repeated many times, and in many different ways. For example, consider the TextRunner Web information extraction system, which extracts relational statements between pairs of entities (e.g., from the phrase "Edison invented the light bulb," TextRunner extracts the relational statement

`Invented(Edison, light bulb)`). In an experiment with a set of about 500 million Web pages, ignoring the extractions occurring only once (which tend to be errors), TextRunner extracted 829 million total statements, of which only 218 million were unique (on average, 3.8 repetitions per statement). Well-known facts can be repeated many times. According to a major search engine, the Web contains over 10,000 statements that Thomas Edison invented the light bulb, and this fact is expressed in dozens of different ways.

This thesis shows that a probabilistic model of the KnowItAll hypothesis, coupled with the redundancy of the Web, can power UIE for arbitrary target concepts. The three primary contributions are discussed below.

1.0.1 Classification with Monotonic Features

The KnowItAll hypothesis can be expressed as an instance of a general problem structure found in many textual classification tasks. Consider a classification task in which each example to be classified is described by a set of features, and we are given the identity of a *monotonic feature* (MF)— a feature whose value is known to vary monotonically with the probability that an example is of a particular class. The KnowItAll hypothesis holds because the textual patterns used for extraction are MFs. For example, the number of occurrences of the phrase "cities such as x" is a monotonic feature for the `City` class.

Monotonic features are readily found in other applications as well. In document classification, the number of times a topic word (e.g. "politics") appears in a document is an MF for the class of documents relevant to the topic. Similar techniques have also been demonstrated to be effective for named-entity recognition, using a set of manually-specified MFs [14]. Manually-specified MFs have also been employed for word-sense disambiguation [59]; this work was later extended to automatically derive the MFs from resources like WordNet [39].

This thesis formalizes the Monotonic Feature Model (MFM) for classification, illustrates several applications in which the model applies, and provides theorems establishing a number of the model's desirable properties. When a single feature exhibiting the monotonicity property is known, PAC learnability is guaranteed from unlabeled data alone, under certain assumptions. The MF model is also formally distinct from, and complementary to, problem structures considered previously in semi-supervised learning. Further, in many cases MFs are more informative relative to labeled examples as the feature space increases in size. This suggests that, as classification tasks trend toward larger and more complex feature spaces, MF-based approaches will become increasingly attractive relative to the standard technique of labeling examples.

In addition to the theoretical analysis, we investigate the power of MFs empirically with experiments in document classification on the standard "20 Newsgroups" data set. A learner given an MF and unlabeled data achieves accuracy equal to that of a state-of-the-art semi-supervised learner relying on 160 hand-labeled examples. Even when MFs are not given as input, their presence or absence can be determined from a small amount of hand-labeled data, which yields a new semi-supervised learning method that reduces error by 15% on the 20 Newsgroups data [19].

1.0.2 The Urns Model of Redundancy in Text

While the notion of a monotonic feature is powerful, and provides performance advantages for document classification, it has important limitations. In more challenging classification settings (including UIE), merely knowing that a feature is monotonic is *not* sufficient to provide accurate classification. We demonstrate this fact formally and empirically, with experiments in UIE.

In order to guarantee high classification accuracy from unlabeled data alone for more challenging tasks, a learner requires more precise information about how target class probability varies with the MF value. In IE, the KnowItAll hypothesis states that the probability that an extraction is correct increases with its repetition. But by how much? How can we precisely quantify our confidence in an extraction given the available textual evidence?

We present an answer to these questions in the form of the Urns model: a special case of the MFM that captures a stronger distributional structure often exhibited by monotonic features in practice. Urns applies to MFs that, like the examples given for IE and document classification above, represent *counts* of repeated observations. Intuitively, Urns considers each observation to be generated by selecting an example from one of two sets: target elements and errors. The observation counts are monotonic features because target elements tend to have higher probabilities of being selected than do error elements.

The formal Urns model is an instance of a classic "balls and urns" model from combinatorics. The MF values represent counts of draws from an urn, where each ball in the urn is tagged with the identity of an example to be classified, and tags can be repeated on different numbers of balls. Given the frequency distribution for tags in the target set and error set, we can compute the probability that an observed example is a target element based how many times its tag is drawn from the urn. A key insight of Urns is that when the frequency distributions have predictable structure (for example, in textual applications the distributions tend to the Zipfian), they can be estimated from unlabeled data alone.

We prove that when the frequency of each tag in the urn is drawn from a mixture of two Zipfian distributions (one for the target class and another for errors), the parameters of Urns can be learned without labeled data. When the data exhibits a certain separability criterion, PAC learnability is guaranteed from unlabeled data alone. We also demonstrate that Urns is effective in practice. In experiments with UIE on the Web, the probabilities produced by the model are shown to be 15 times better, on average, when compared with techniques from previous work [20].

1.0.3 Language Models for Assessing Sparse Extractions

Because extraction frequency follows a Zipf distribution, even on the Web a substantial fraction of extractions appear infrequently. The KnowItAll hypothesis alone is ineffective at assessing which of these *sparse* extractions are correct. In fact, over 50% of the extractions in our experiments appear only once, meaning that redundancy-based methods such as Urns have no way of assessing which extraction is more likely to be correct for fully half of the extractions.

We show that it is possible to assess sparse extractions by leveraging the KnowItAll hypothesis in conjunction with the distributional hypothesis of language. Namely, sparse extractions which appear in contexts more similar to those of common extractions are more likely to be correct.

We introduce the insight that the sub-field of statistical language modeling provides unsupervised methods that can be leveraged to assess sparse extractions. In experiments in UIE, an assessment technique based on language models is shown to reduce extraction error by 39%, on average, when compared with techniques from previous work. Further, because unsupervised language models can be pre-computed without any labeled data or seed examples, they are ideally suited to UIE and are substantially more scalable than techniques from previous work [21].

1.1 Thesis Outline

We present the Monotonic Feature Model in Chapter 2, including theoretical results and experiments demonstrating its efficacy in document classification. In Chapter 3, we highlight the limitations of the MFM and introduce the Urns model. We present theoretical results and experiments in UIE. We show how unsupervised language models can be used to assess sparse data in Chapter 4. Chapter 5 concludes with a discussion of future work.

Chapter 2

CLASSIFICATION WITH MONOTONIC FEATURES

2.1 Introduction

Is accurate classification possible in the complete absence of hand-labeled data? *A Priori*, the answer would seem to be no, unless the learner has knowledge of some additional problem structure. This chapter identifies a problem structure, called *Monotonic Features (MFs)*, that enables the learner to automatically assign probabilistic labels to data. A feature is said to be *monotonic* when the probability of class membership increases monotonically with that feature's value, all else being equal.

MFs occur naturally in a broad range of textual classification tasks. For example, it can be shown that Naive Bayes text classifiers return probability estimates that are monotonic in the frequency of a word—for the class in which the word is most common. Thus, if we are trying to discriminate between documents about New York and Boston, then we expect to find that the Naive Bayes feature measuring the frequency of "Giants" in the corpus is an MF for the class New York, and likewise for "Patriots" and Boston.

As argued in the introduction, MFs are readily apparent in Information Extraction (IE) where strings are extracted from sentences and classified into categories (*e.g.* `City` or `Film`) based on their proximity to "extraction patterns". For example, the phrase "cities such as" is an extraction pattern. Any proper noun immediately following this pattern is likely to denote a city, as in the phrase "cities such as Boston, Seattle, and New York" [33]. When classifying a proper noun, the KnowItAll hypothesis asserts that the *number of times* the proper noun follows an extraction pattern in a corpus turns out to be a powerful MF.

The power of MFs is not restricted to information extraction. In document classification, the name of the class is a natural MF—the more times it is repeated in a document, all other things being equal, the more likely it is that the document belongs to the class. We demonstrate this to be the case empirically in Section 2.5, extending the experiments of

[30] and [11]. Similarly, information retrieval systems classify documents into relevant and irrelevant documents based, in part, on the Term-Frequency-Inverse-Document-Frequency (TF-IDF) metric, and then proceed to rank the relevant documents. The term frequency component of this metric is a monotonic feature. MF-based techniques have been demonstrated to be effective for named-entity recognition, using a set of manually-specified MFs [14]. Manually-specified MFs have also been employed for word-sense disambiguation [59]; this work was later extended to automatically derive the MFs from resources like WordNet [39].

Thus, MFs have been used *implicitly* in a broad range of textual classification tasks. This chapter makes the MF abstraction *explicit*, providing a formal theory of MFs. We also devise an automatic method for explicitly detecting and utilizing MFs, and quantify the method's benefits empirically.

2.1.1 Contributions

Typically, MFs cannot serve directly as classifiers. Instead, this chapter presents theoretical and empirical results showing that even relatively weak MFs can be used to induce a noisy labeling over examples, and these examples can then be used to train effective classifiers utilizing existing supervised or semi-supervised techniques.

Our contributions are as follows:

1. We prove that the Monotonic Feature (MF) model guarantees PAC learnability using only unlabeled data, and that MFs are distinct from and complementary to standard biases used in semi-supervised learning, including the manifold and cluster assumptions. Further, when compared with labeled examples, MFs provide relatively greater information gain about unlabeled examples' labels as the feature set increases in size.

2. We present a general technique, called Mfa, for employing MFs in combination with an arbitrary concept learning algorithm. We demonstrate experimentally that Mfa can outperform state-of-the-art techniques for semi-supervised document classification, including Naive Bayes with Expectation Maximization (NB-EM), and Label Propagation, on the 20 Newsgroups data set [41].

The remainder of the paper is organized as follows. Section 2.2 defines the classification task we consider, and Section 2.3 defines the formal properties of monotonic features. Section 2.4 presents our theoretical results, and formalizes the relationship between the MF approach and previous work. We present experimental results in Section 2.5, and conclude with directions for future work.

2.2 Formal Problem Definition, Conventions and Notations

We consider a semi-supervised classification task, in which the goal is to produce a mapping from an instance space $\mathcal{X}$ consisting of d-tuples $\mathbf{x} = (x_1, \ldots, x_d)$, to a binary output space $\mathcal{Y} = \{0, 1\}$.[1] We denote the concept class of mappings $f : \mathcal{X} \to \mathcal{Y}$ as $\mathcal{C}$.

We assume the following inputs:

- A set of zero or more *labeled* examples $D_L = \{(\mathbf{x}_i, y_i) | i = 1 \ldots m\}$, drawn i.i.d. from a distribution $P(\mathbf{x}, y)$ for $\mathbf{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$.

- A set of zero or more *unlabeled* examples $D_U = \{(\mathbf{x}_i) | i = 1 \ldots u\}$ drawn from the marginal distribution $P(\mathbf{x}) = \sum_y P(\mathbf{x}, y)$.

- A set $M \subset \{1, \ldots, d\}$ of zero or more *monotonic features* for the positive class $y = 1$. The monotonic features have properties specified below.

We further define $\mathcal{C}_M \subset \mathcal{C}$ as the concept class of binary classifiers that use only the monotonic features. Similarly, let $\mathcal{C}_{\neg M} \subset \mathcal{C}$ indicate the concept class of binary classifiers using only the non-monotonic features.

The goal of the task is to produce a mapping $c \in \mathcal{C}$ that maximizes classification accuracy $acc(c)$ evaluated over a set of test examples D_T also drawn i.i.d. from $P(\mathbf{x}, y)$:

$$acc(c) = \frac{1}{|D_T|} \sum_{(\mathbf{x},y) \in D_T} \delta(c(\mathbf{x}), y)$$

where δ represents the Kronecker delta, equal to one when its arguments are equal and zero otherwise.

[1]For convenience, we restrict our formal framework to the binary case, but the techniques and analysis can be extended trivially to the multi-class case.

We will also define the *error* of a classifier $c \in \mathcal{C}$ as

$$error(c) = 1 - acc(c)$$

We will occasionally refer to the error over a particular set of examples S, indicated as $error_S(c)$.

In addition to evaluating classifiers in terms of accuracy, in Chapter 3 we will also evaluate probabilistic models $P_{model}(y|\mathbf{x})$, which explicitly estimate probabilities of class membership based on the inputs. The probabilistic models are evaluated in terms of conditional log likelihood score $ll(model)$, defined as:

$$ll(model) = \sum_{(\mathbf{x},y) \in D_T} \log P_{model}(y|\mathbf{x}) \tag{2.1}$$

2.3 Definition of Monotonic Features

Monotonic features exhibit a monotonically increasing relationship with the probability that an example is a member of the positive class. More formally, we define monotonic features as follows:

Definition 1 *A* **monotonic feature** *for class y is a feature $i \in \{1, \dots, d\}$ for which the following three properties hold:*

- *The domain of x_i is fully ordered and discrete, and has finite support.*[2]
- *The conditional probability that an example is an element of class $y = 1$ increases strictly monotonically with the value of x_i. That is, $P(y = 1|x_i = r) > P(y = 1|x_i = r')$ if $r > r'$.*
- *The monotonicity is* non-trivial *in that $P(x_i)$ has positive probability for more than one feature value. That is, there exists $r > r'$ and $\epsilon > 0$ such that $P(x_i = r), P(x_i = r') > \epsilon$.*

With this definition, we can state precisely the monotonic feature model:

[2]For convenience, we present our analysis in terms of discrete and finite monotonic features, but the results can be extended naturally to the continuous case.

Definition 2 *For a learning problem from the input space $\mathcal{X}$ of d-tuples $\mathbf{x} = (x_1, \ldots, x_d)$ to the output space $\mathcal{Y}$, the* **monotonic feature model (MFM)** *holds if and only if at least one of the features $i \in \{1, \ldots, d\}$ is a monotonic feature for the positive class $y = 1$.*

When tasked with a learning problem for which the MFM holds, three distinct configurations of the input are possible. First, monotonic features may be known in the absence of labeled data ($|M| > 0, D_L = \emptyset$). This is the setting considered in previous applications of monotonic features, as discussed in Section 2.1. Second, monotonic features may be unknown, but labeled data may be provided ($M = \emptyset, |D_L| > 0$); this corresponds to standard semi-supervised learning. In this case, the MFM can still be exploited by identifying monotonic features using the labeled data, as described in Section 2.5.1. Lastly, both monotonic features and labeled data may be provided ($|M|, |D_L| > 0$). We provide algorithms for each case and evaluate each experimentally in Section 2.5.

2.4 Theory of Monotonic Features

This section shows that under certain assumptions, knowing the identity of a single monotonic feature is sufficient to PAC learn a target concept from *only unlabeled data*. Further, we prove that monotonic features become more informative relative to labeled examples as the feature set size increases. Lastly, we discuss and formally establish distinctions between the monotonic feature abstraction and other semi-supervised techniques.

We start by introducing the *conditional independence* assumption, which states that the monotonic features are conditionally independent of the non-monotonic features given the class. Formally, the conditional independence assumption is satisfied *iff* $P(\{x_i : i \in M\}|y, \{x_j : j \notin M\}) = P(\{x_i : i \in M\}|y)$.

We show that when the concept class $\mathcal{C}_{\neg M}$ is learnable in the PAC model with classification noise, and the conditional independence assumption holds, then knowledge of a single monotonic feature makes the full concept class $\mathcal{C}$ learnable from *only* unlabeled data.

Our result builds on the result stated below (adapted from [8]). The theorem requires the following definition:

Definition 3 *A classifier $h \in C_M$ is* **weakly-useful** *iff it has the following two properties* *for $\epsilon > 0$:*

1. $P(h(\mathbf{x}) = 1) \geq \epsilon$, *and*

2. $P(y = 1 | h(\mathbf{x}) = 1) \geq P(y = 1) + \epsilon$.

Theorem 4 *(adapted from [8]) If the conditional independence assumption is satisfied, and $\mathcal{C}_{\neg M}$ is learnable with classification noise, then $\mathcal{C}$ is learnable using unlabeled data alone, given a classifier $h \in C_M$ which is weakly useful with probability $1 - \delta$ for all $\delta > 0$.*

The original theorem in [8] assumes that a weakly-useful classifier is given, but it is straightforward to show that $\mathcal{C}$ is PAC learnable if h has the weakly-useful properties only with probability $1 - \delta$ for all $\delta > 0$.[3] We can now state our result.

Corollary 4.1 *If the conditional independence assumption is satisfied and the concept class $C_{\neg M}$ is learnable in the PAC model with classification noise, then given a single monotonic feature, C is learnable from unlabeled data only.*

Proof. Our proof proceeds by showing that for any $\delta > 0$, with probability $1 - \delta$ a single monotonic feature $i \in M$ is sufficient to produce a classifier h with the two weakly useful properties. First, by assumption we know there exist distinct r, r' such that $P(x_i = r), P(x_i = r') > \epsilon'$. Thus, the probability that in m unlabeled examples we observe at least two distinct values for x_i, and each at least $m\epsilon'/2$ times, is at least $4e^{-m(\epsilon'/2)^2}$ by Hoeffding's inequality and a union bound over the two values. Given that we do observe at least two values with sufficient probability, we construct a single-threshold classifier h that partitions the feature x_i such that at least $m\epsilon'/2$ of the m unlabeled samples fall on either side of the classification threshold. Then again using a Hoeffding inequality, we know that with probability $2e^{-2m(\epsilon'/4)^2}$, the probability mass on each side of the classifier's threshold is at least $\epsilon'/4$. In this case, the classifier h meets the required properties, since $P(h(\mathbf{x}) = 1) \geq \epsilon'/4$, and by the monotonicity property of the monotonic feature, $P(y =$

[3]In this theorem and PAC learnability results that follow, we are proving only that a Probably Approximately Correct classifier can be obtained from unlabeled data, not that such a classifier can be found using a polynomial number of examples or in polynomial time. In this particular case, the latter, stronger result holds for Corollary 4.1 whenever it holds for Theorem 4.

$1|h(\mathbf{x}) = 1) \geq P(y = 1) + \epsilon$. Using a union bound, we can produce such a classifier h with probability at least $1 - \delta$ using

$$m > \frac{K}{\epsilon'^2 \delta} \tag{2.2}$$

unlabeled examples for K a constant. □

The next theorem suggests that monotonic features become preferable to labeled examples as the feature space increases in size. We compare the value of monotonic features and labeled examples in terms of information gain, defined below. For convenience, these results are presented using a feature space $\mathcal{X}_B$, in which all features are binary-valued.

Definition 5 *The* **information gain** *with respect to an unlabeled example's label y provided by a variable v is defined as the reduction in entropy of y when v is given, that is $\sum_{y'=0,1} P(y = y'|v) \log P(y = y'|v) - P(y = y') \log P(y = y')$.*

Next, we define the two properties of the classification task that our theorem requires. Informally speaking, the first property states that the feature space does not have fully redundant features, whereas the second states that examples which are far apart have less dependent labels than those which are close together. We would expect these properties to hold for most tasks in practice.

Definition 6 *A distribution $\mathcal{D}$ on $(\mathcal{X}_B, \mathcal{Y})$ has* **bounded feature dependence** *if there exists $\epsilon_F > 0$ such that the conditional probability $P_{\mathcal{D}}(x_i = r|\{x_j = r_j : j \neq i\}) < 1 - \epsilon_F$ for all i, r, and and sets $\{x_j = r_j : j \neq i\}$ of assignments to one or more x_j.*

Definition 7 *A distribution $\mathcal{D}$ on $(\mathcal{X}_B, \mathcal{Y})$ has* **distance-diminishing information gain** *if the information gain of an example $\mathbf{x}$ with respect to the label of any neighboring example $\mathbf{x}'$ is less than $K_I \delta_I^r$ for $\delta_I < 1$, where r is the Hamming distance between $\mathbf{x}$ and $\mathbf{x}'$.*

The following theorem shows that whenever the above properties hold to a sufficient degree, the expected information gain from a labeled example falls as the size of the feature space increases.

Theorem 8 *For a learning problem governed by distribution $\mathcal{D}$ with bounded feature dependence and distance-diminishing information gain, with $\epsilon_F > \frac{\delta_I}{\delta_I + 1}$, as the number of features*

d increases, the expected information gain provided by a labeled example about unlabeled examples' labels decreases to zero. However, the information gain from an MF x_f with given relationship $P(y|x_f)$ remains constant as d increases.

Proof. With bounded feature dependence, we have for all $\mathbf{x} \in \mathcal{X}_B$:

$$P(\mathbf{x}) = \prod_{i=0}^{d} P(x_i|\{x_{i+1}, \ldots, x_d\}) < (1 - \epsilon_F)^{d-1} \tag{2.3}$$

For a given labeled example, the number of examples $\mathbf{x} \in \mathcal{X}_B$ at Hamming distance r is $\binom{d}{r}$. Thus, the expected number of unlabeled examples at distance r is less than $u(1-\epsilon_F)^{d-1}\binom{d}{r}$. Because $\mathcal{D}$ has distance-diminishing information gain, the expected information gain from a labeled example about neighboring examples is less than $\sum_r K_I \delta_I^r u(1-\epsilon_F)^{d-1}\binom{d}{r}$. Using the fact that $\sum_r \delta_I^r \binom{d}{r} = (1+\delta_I)^d$, we can re-write the expected information gain as $u((1+\delta_I)(1-\epsilon_F))^{d-1}K_I(1+\delta_I)$. Then with the assumption that $\epsilon_F > \frac{\delta_I}{\delta_I+1}$, we have that $(1+\delta_I)(1-\epsilon_F) < 1$ and thus the expected information gain of a labeled example tends to zero as d increases. Finally, note that the information provided by an MF x_f about unlabeled examples' labels depends only on the given relationship $P(y|x_f)$, and not on d. Thus, this information gain is constant with d. $\square$

The portion of Theorem 8 which concerns information gain of a labeled example is a version of the well-known "curse of dimensionality" [7], which states that the number of examples needed to estimate a function scales exponentially with the number of dimensions under certain assumptions. Theorem 8 differs in detail, however; it states the curse of dimensionality in terms of information gain, making possible a direct comparison with monotonic features.

2.4.1 Relation to Other Approaches

In Section 2.1, we identified several learning methods that utilized Monotonic Features (MFs) implicitly, which was a key motivation for formalizing MFs. This section explains the ways in which MF-based classification is distinct from previous semi-supervised learning methods.

When MFs are provided as input, they can be viewed as a kind of "labeled feature" studied in [22]. However, instead of "expectation regularization", we will use the MFs to probabilistically label examples. Thus, MFs can complement any concept learning algorithm, not just discriminative probabilistic models as in [22]. Moreover, in Section 2.5, we show that MFs can either obviate hand-labeled data, or can be estimated automatically from a small number of hand-labeled samples.

Co-training [8] is a semi-supervised technique that also considers a partition of the feature space into two distinct "views." One might ask if monotonic feature classification is equivalent to co-training with the monotonic features serving as one view, and the other features forming the other. However, co-training requires labeled data to train classifiers for each view, unlike monotonic feature classification which can operate without any labeled data. Thus, there are cases where an MF-based algorithm like Mfa is applicable, but co-training is not.

Even when labeled data is available, co-training takes the partition of the feature set as input, whereas monotonic features can be detected automatically using the labeled data. Also, co-training is an iterative algorithm in which the most likely examples of a class according to one view are used to train a classifier on the other view in a mutually recursive fashion. For a given set of monotonic features, however, iterating this process is ineffective, because the mostly likely examples of a class according to the monotonic feature view are fixed by the monotonicity property.

The area of probabilistic IR (*e.g.*, [15]) studies models for determining the probability that a document is relevant to a particular query, based on features like the frequency of the query in the document. Our work is similar in the sense that Mfa uses occurrences of the class name to classify documents, but is distinct in important ways. Unlike probabilistic IR, we pose a general framework for utilizing monotonic features and existing semi-supervised algorithms to perform classification. Also, the target of Mfa is classification, rather than ranking as is typical in probabilistic IR. Partly because of this, Mfa invokes an underlying semi-supervised algorithm which learns parameters for each class. This is distinct from probabilistic IR, which focuses on optimizing a fixed function that can be used to rank documents for any given query.

A similar insight to the MFM is inherent in the recent Dataless Classification model [11]. In that work, documents are classified by estimating their "semantic similarity" to the class name. [11] shows that classification accuracy can be improved by computing semantic similarities with the aid of an external knowledge source (Wikipedia). These similarity scores can be viewed as MFs. In contrast to [11], our focus is on the monotonic feature abstraction, which arises in a number of distinct tasks in addition to document classification, as discussed in Section 2.1. We provide general techniques for utilizing the MFM in classification, along with theorems establishing the model's key properties.

Semi-supervised Smoothness Assumptions

In this section, we prove that the MFM is distinct from certain smoothness properties typically assumed in previous semi-supervised learning methods, known as the cluster and manifold assumptions. The *cluster assumption* states that in the target concept, the boundaries between classes occur in relatively low-density regions of the distribution $P(\mathbf{x})$. The *manifold assumption* states that the distribution $P(\mathbf{x})$ is embedded on a manifold of strictly lower dimension than the full input space $\mathcal{X}$.

It can be shown that classification tasks exhibiting the MFM exist for which neither the cluster assumption nor the manifold assumption holds. Similarly, we can construct classification tasks exhibiting the manifold assumption, the cluster assumption, or both, but without the MFM. We demonstrate this formally below.

In the below theorems, we will consider an input space $\mathcal{X}_0$ of vectors $\mathbf{x} = (x_1, \ldots, x_d)$ where x_1 is an integer in the interval $[0, K]$ and $(x_2, \ldots, x_d)$ is a vector of real numbers in $[0, K]^{d-1}$, for K an integer greater than 1.

Theorem 9 *The monotonic feature model does not imply either the manifold assumption or the cluster assumption.*

Proof. The proof is by construction of a distribution on $(\mathcal{X}_0, \mathcal{Y})$ for which the monotonic feature model holds, but neither the manifold nor cluster assumptions hold. Define a distribution $\mathcal{D}$ over $(\mathcal{X}_0, \mathcal{Y})$ for which the marginal distributions $P_{\mathcal{D}}(y = 0, x_1 = r) \propto 1 - r/K^2$ and $P(y = 1, x_1 = r) \propto 1 - (r/K - 1)^2$, and assume that the distribution $P_{\mathcal{D}}(x_2, \ldots, x_d, y)$

is uniform. It can be shown that $P_{\mathcal{D}}(y=1|x_1=r) > P_{\mathcal{D}}(y=1|x_1=r-1)+\epsilon$ for $\epsilon = \frac{1}{3K}$, and that $P_{\mathcal{D}}(x_1=0), P_{\mathcal{D}}(x_1=1) > \epsilon'$ for $\epsilon' = \frac{1}{2K}$; thus, x_1 is a monotonic feature for the class $y=1$. However, by construction the distribution $P(\mathbf{x})$ is fully dense in the space $\mathcal{X}_0$, so the manifold assumption does not hold. Additionally, the cluster assumption does not hold, as the optimal classification boundary between $y=0$ and $y=1$ is the hyperplane through $x_1 = K/2$, the most dense portion of $P_{\mathcal{D}}(\mathbf{x})$. $\square$

Theorem 10 *The monotonic feature model is not implied by either the manifold assumption, the cluster assumption, or their conjunction.*

Consider a distribution $\mathcal{D}$ on $(\mathcal{X}_0, \mathcal{Y})$ with marginal distribution $P_{\mathcal{D}}(y=0, x_1=r) \propto 1$ for $r < K/5$ and $r > 4K/5$, and $P_{\mathcal{D}}(y=1, x_1=r) \propto 1$ for $2K/5 < r < 3K/5$. Let $P_{\mathcal{D}}(x_1)$ be zero otherwise. In this case, the class boundaries lie within a region of zero density, so the cluster assumption holds. Further, we can select whether or not the manifold assumption holds by adjusting the remainder of the distribution. Specifically, if we set $P_{\mathcal{D}}(x_2, \ldots, x_d)$ to be one *iff* $(x_2, \ldots, x_d)$ is the zero vector, then the manifold assumption holds, whereas if $P_{\mathcal{D}}(x_2, \ldots, x_d)$ is uniform, the manifold assumption does not hold. In either case, $P_{\mathcal{D}}(y=1|x_i)$ cannot be monotonic for any i, meaning that the monotonic feature model does not hold. $\square$

In conclusion, it is clear that the MFM is a novel problem structure that is both conceptually and formally distinct from previously articulated structures including co-training, the cluster assumption, and the manifold assumption.

2.5 Experiments

This section reports on our experiments in utilizing MFs for text classification. As discussed in Section 2.1, MFs have been used *implicitly* by several classification methods in numerous tasks. Here we quantify their impact on the standard "20 newsgroups" data set [41]. We show that MFs can be employed to perform accurate classification even without labeled examples, extending the results from [30] and [11] to a semi-supervised setting. Further, we also demonstrate that whether or not the identities of MFs are given, exploiting the MF model by learning MFs can improve performance.

We begin by defining a set of methods for employing MFs in general classification settings.

2.5.1 General Methods for Monotonic Feature Classification

Here, we define a set of abstract methods for incorporating monotonic features into any existing learning algorithm. The first method, Mfa, is an abstraction of the MF word sense disambiguation algorithm first introduced in [59]. It is applicable when monotonic features are given but labeled examples are not provided. The second, Mfa-ssl , applies in the standard semi-supervised learning case when some labeled examples are provided, but the identities of the MFs are unknown and must be learned. Lastly, Mfa-both applies when both labeled data and MF identities are given.

When monotonic features are given to the algorithm without labeled data, Mfa proceeds as shown in Figure 2.1. Mfa labels the unlabeled examples D_U as elements of class $y = 1$ *iff* some monotonic feature value x_i for $i \in M$ exceeds a threshold τ. The threshold is set using unlabeled data so as to maximize the minimum probability mass on either side of the threshold.[4] This set of bootstrapped examples D'_L is then fed as training data into a supervised or semi-supervised algorithm $\Phi(D'_L, D_U)$, and Mfa outputs the resulting classifier. In general, the Mfa schema can be instantiated with any machine learning algorithm Φ.

When MFs are unknown, but some labeled data is given, the Mfa-ssl (Figure 2.2) algorithm attempts to identify MFs using the labeled training data D_L, and adds the most strongly monotonic features to the set M. Monotonicity strength can be measured in various ways; in our experiments below, we rank each feature x_i by the quantity $f(y, x_i) = \sum_r P(y, x_i = r)r$ for each class y. Note that this expression requires that $f(y, x_i)$ have a numerical interpretation; when the MF is merely ordinal, other monotonicity measures (e.g., rank correlation) would be required. Mfa-ssl then adds monotonic features to M in descending order of this value, up to a limit of $k = 5$ per class.[5]

[4]This policy is suggested by the proof of Corollary 4.1, in which the only requirement of the threshold is that sufficient mass lies on each side.

[5]A sensitivity analysis revealed that varying k by up to 40% in either direction did not decrease perfor-

```
MFA($M$, $D_U$, $\Phi$)
    1. $D'_L$ = Labeled examples $(\mathbf{x}, y)$ such that $y = 1$
       *iff* a $x_i > \tau$ for some $i \in M$
    2. Output $\Phi(D'_L, D_U)$
```

Figure 2.1: Pseudocode for MFA. The inputs are M, a set of monotonic features, D_U, a set of unlabeled examples, and $\Phi(L, U)$, a supervised or semi-supervised machine learning algorithm which outputs a classifier given labeled data L and unlabeled data U. The threshold τ is derived from the unlabeled data and M (see text).

MFA-SSL then invokes a given machine learning algorithm $\Phi_M(M, D_L, D_U)$ to learn a probabilistic classifier that employs *only* the monotonic features in M. MFA-SSL uses this classifier to probabilistically label the examples in D_U to form D'_L. MFA-SSL then returns $\Phi(D'_L, D_U)$ as in MFA. A feature of MFA-SSL is that if no monotonic features are identified, MFA-SSL defaults to the underlying algorithm Φ.

For small amounts of labeled data, random variation on individual examples can lead to inaccurate estimates for $f(y, x_i)$; we take three steps to help ensure that our estimates are robust. First, we smooth the estimates using a Laplacian prior equivalent to the observation of half of a negative example.[6] Second, we do not add features to M for a class unless they occur for more than one distinct example of the class in the training set. Third, we estimate the $f(y, x_i)$ value as the average of the maximum a posteriori estimates computed via leave-one-out sub-sampling.

When monotonic features are known *and* labeled examples are available, we run a derivative of MFA-SSL denoted as MFA-BOTH. The algorithm is the same as MFA-SSL, except that any given monotonic features are added to the learned set in Step 1 of Figure 2.2, and bootstrapped examples using the given monotonic features (from Step 1 in Figure 2.1) are added to D'_L.

mance of MFA-SSL in the experiments in Section 2.5.3.

[6]A sensitivity analysis revealed that varying the prior weight by 50% in either direction changed average performance by less than 0.1% in the experiments in Section 2.5.3.

```
MFA-SSL($D_L$, $D_U$, $\Phi$, $\Phi_M$)
  1. $M$ = the $k$ strongest monotonic features in $D_L$
  2. $D'_L$ = Examples from $D_U$ probabilistically
     labeled with $\Phi_M(M, D_L, D_U)$
  3. Output $\Phi(D_L \cup D'_L, D_U)$
```

Figure 2.2: Pseudocode for MFA-SSL. The inputs D_U and Φ inputs are the same as those of MFA (see Figure 2.1). The additional inputs include labeled data D_L and a machine learning algorithm $\Phi_M(M, L, U)$ which given labeled data L and unlabeled data U outputs a probabilistic classifier that uses only monotonic features M. k is a parameter of MFA-SSL(see text).

2.5.2 Experimental Methodology and Baseline Methods

We evaluate the monotonic feature approach on two tasks: document classification and information extraction.

For document classification, we employ the standard 20 newsgroups data set; the goal of the task is to determine from the text of a newsgroup post the newsgroup in which it appeared. We used bag-of-word features after converting terms to lowercase, discarding the 100 most frequent terms and all terms appearing only once. Below, we present results averaged over four disjoint training sets of variable size, using a disjoint test set of 5,000 documents and an unlabeled set of 10,000 documents.

When the identities of monotonic features are given, we obtained one-word monotonic features simply using the newsgroup name, with minor modifications to expand abbreviations. This methodology closely followed that of [30]. For example, the occurrence count of the term "politics" was a monotonic feature for the `talk.politics.misc` newsgroup. We also expanded the set of monotonic features to include singular/plural variants.

The second task we investigate is information extraction. Here, we use a data set from previous work, which includes potential extractions from three classes (`City, Country, Film`) [25]. We treated this as a four-valued classification task, with errors of any type forming the fourth class. The original data set also includes a monotonic feature for each extraction: the number of times each extraction appeared in a search of the Web for extraction patterns based on the class name (e.g., "cities such as <x>"). To generate additional features, we obtained another corpus of 12,076,276 distinct sentences from the Web, and created a feature set for each example consisting of its co-occurrence counts with each distinct surrounding context of two to four words.[7]

We considered only those extractions that appeared at least once in our corpus. Our final data set included 1020 labeled examples, and 2998 unlabeled examples. Unlike the 20 newsgroups data set, the extraction data set is highly unbalanced; the labeled set includes 301 `City` extractions, 56 `Country` extractions, 206 `Film` extractions, and 457 errors. We discarded the 1000 most frequent contexts, and also discarded any contexts which did not appear with at least 10 distinct extractions from the data set, resulting in a feature set of 191,724 distinct contexts. In each extraction experiment, we present the average performance over eight random cross-validation runs.

Baseline methods

We compared the monotonic feature approach with two alternative algorithms, which represent two distinct points in the space of semi-supervised learning algorithms. The first, **NB-EM**, is a semi-supervised Naive Bayes with Expectation Maximization algorithm [43]. For NB-EM, we used the settings shown to perform well for the 20 newsgroups data set in [43], including normalizing feature vectors by document length. [8]

The second algorithm, **LP**, is a semi-supervised graph-based label propagation algorithm recently employed for text classification [12]. We used a cosine metric for distance between examples. LP requires a function to convert these distances into weights—we found that the function given in [12] was not effective for our data sets. Instead, we used the following equation to convert the cosine distance $d_{i,j}$ between two examples into a weight $w_{i,j}$:

$$w_{i,j} = e^{\frac{1}{d_{i,j}-1}} \tag{2.4}$$

[7]The ground truth and corpus used in these experiments is available for download – see Appendix A.

[8]We ran the EM algorithm for 10 iterations; experiments showed that increasing the number of iterations to 20 had negligible effects on performance.

Strictly speaking, LP is a *transductive* classification algorithm; it requires that test examples be present during training. To evaluate LP in our semi-supervised setting, in which test examples are not available during training, we first ran the LP algorithm using the training and unlabeled examples.[9] Then, for each test example, we output a classification by propagating labels from the training and unlabeled examples to the test example.

For simplicity, we first evaluate our two baseline algorithms on each data set; we then compare the monotonic feature approach relative to the best performing baseline in each case. Figure 2.3 shows that the NB-EM algorithm substantially outperforms LP on the 20 newsgroups data set. However, Figure 2.4 shows that LP outperforms NB-EM on the extraction task.[10] Thus, we compare with NB-EM for the document classification task, and LP for the extraction task.

In each experiment, for the monotonic feature approach we set the underlying algorithm Φ to the baseline algorithm used for comparison. In NB-EM, we weighted the set of examples labeled using the monotonic features (D'_L) equally with the original labeled set, increasing the weight by the equivalent of 200 labeled examples when monotonic features are given to the algorithm. For LP, we weight the examples D'_L evenly with the original labeled set, but use D'_L only to initialize the label propagation and not thereafter. "Clamping" the bootstrapped examples D'_L (as is standard in LP with labeled examples) resulted in the bootstrapped labels overwhelming the algorithm. In all cases, we employed a supervised Naive Bayes classifier for Φ_M.

2.5.3 Experimental Results in Document Classification

The first question we investigate is what level of performance MFA can achieve *without* labeled training data, when monotonic features are given. The results of this experiment are shown in Table 2.1. MFA achieves accuracy on the 20-way classification task of 0.563.

[9]We ran the LP algorithm for 200 iterations; experiments showed that increasing the number of iterations to 500 did not increase performance.

[10]Notice that NB-EM *decreases* in performance given additional training data. This is not solely a consequence of the semi-supervised algorithm, as a supervised NB classifier was found to exhibit the same trend. We attribute this behavior to the many interdependencies in the features in the extraction task, which badly violate the conditional independence assumption of the Naive Bayes classifier.

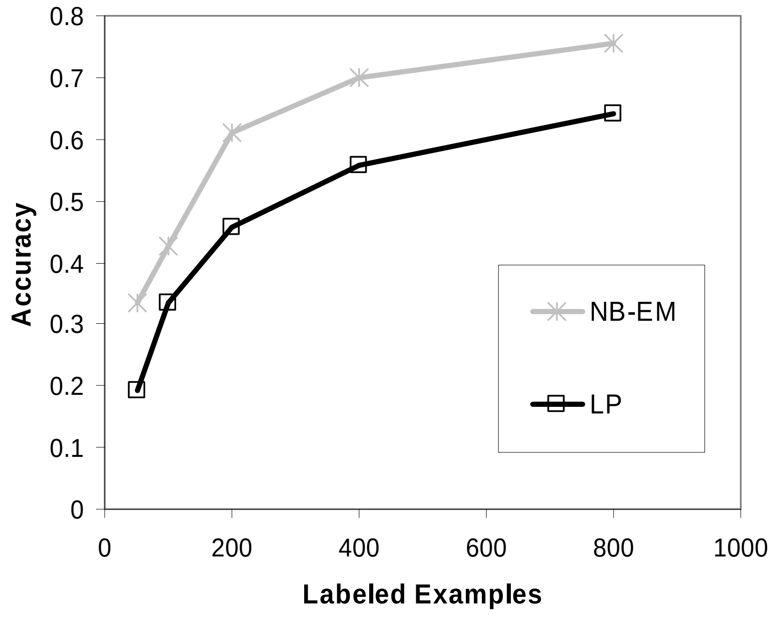

Figure 2.3: Baseline algorithm performance in document classification. The NB-EM baseline outperforms LP.

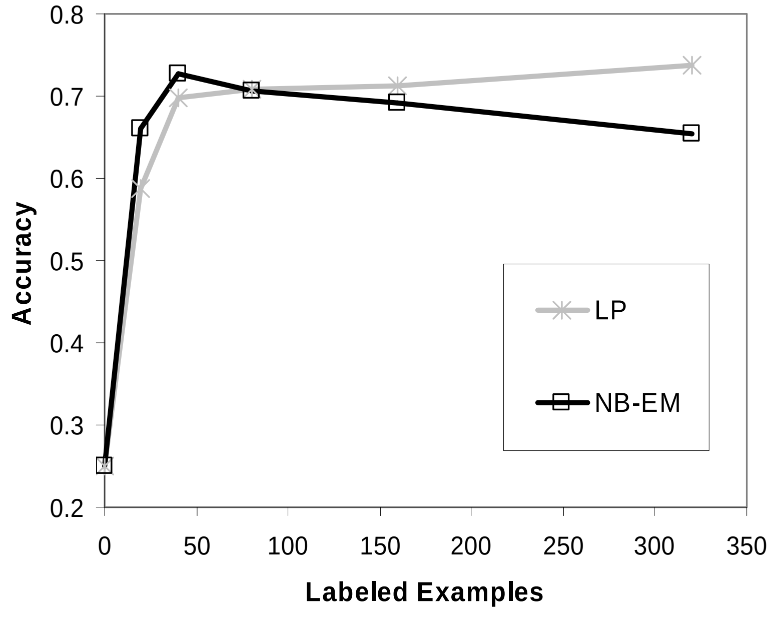

Figure 2.4: Baseline algorithm performance in extraction. LP outperforms NB-EM.

Another way to measure this accuracy is in terms of the number of labeled examples that a baseline semi-supervised technique would require in order to achieve comparable performance. We found that MFA outperformed NB-EM with up to 160 labeled examples. This first experiment is similar to that of [30], except that instead of evaluating against only supervised techniques, we use a more comparable semi-supervised baseline (NB-EM).

Could the monotonic features, on their own, suffice to directly classify the test data? To address this question, the table also reports the performance of using the given monotonic features exclusively to label the test data (MF Alone), without using the semi-supervised technique Φ. We find that the bootstrapping step provides large benefits to performance; MFA has an effective number of labeled examples eight times more than that of MF Alone.

Table 2.1: Performance of MFA on document classification when monotonic features are given, and no labeled examples are provided. MFA achieves accuracy of 0.563, which is ten fold that of a Random Baseline classifier that assigns labels randomly, and more than double that of "MF Alone", which uses only the monotonic features and ignores the other features. MFA's accuracy exceeds that of the NB-EM baseline with 160 labeled training examples, and is eight fold that of "MF Alone".

	Random Baseline	MF Alone	MFA
Accuracy	5%	24%	56% **(2.33x)**
Labeled Example Equivalent	0	20	160 **(8x)**

The second question we investigate is whether the monotonic feature approach can improve performance even when the class name is not given. MFA-SSL takes the same inputs as the NB-EM technique, without the identities of monotonic features. The performance of MFA-SSL as the size of the labeled data set varies is shown in Figure 2.5. The graph shows that for small labeled data sets of size 100-400, MFA-SSL outperforms NB-EM by an average error reduction of 15%. These results are statistically significant ($p < 0.001$, Fisher Exact Test).

Lastly, in the case that both monotonic features and labeled examples are available,

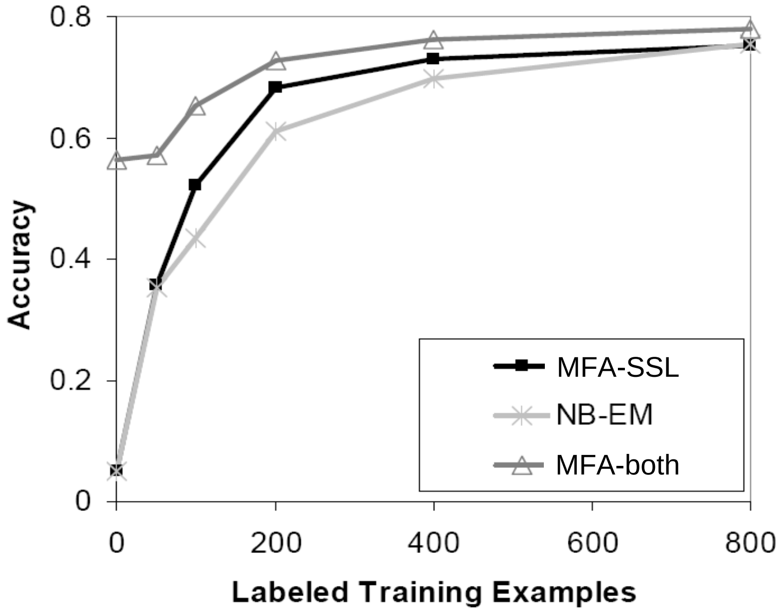

Figure 2.5: Performance in document classification. Mfa-ssl reduces error over the NB-EM baseline by 15% for training sets between 100 and 400 examples, and Mfa-both reduces error by 31% overall.

Mfa-both reduces error over the NB-EM baseline by an average of 31% across the training set sizes shown in Figure 2.5.

2.5.4 Information Extraction Results

We performed the same set of experiments as above in the information extraction domain. In the case in which no labeled examples are provided, the results are shown in Table 2.2. While the monotonic features double classification accuracy over the baseline, this is equivalent to the baseline label propagation algorithm using only eight labeled examples. Further, bootstrapping with additional features does not help performance (in fact, Mfa performs slightly worse than the MF alone).

We expect that the reasons the additional features Mfa uses do not improve performance are twofold. The first is that the extraction task is more difficult than the 20 newsgroups classification task, because the extracted classes are more similar and the problem space contains many overlapping concepts; we show formally how this situation makes learning from MFs more difficult in Section 3.1. Second, the features in the extraction task (contexts surrounding each extraction) have substantial interdependencies, and we expect these mislead the label propagation algorithm and decrease its effectiveness.

Figure 2.6 shows the performance of Mfa-ssl and Mfa-both versus LP in a semi-supervised setting. As the graph shows, while Mfa-both outperforms the other methods when no labeled examples are provided, the MFM offers no consistent performance improvements when labeled examples are present. In Chapter 3, we analyze the limitations of the MFM in this domain, and describe a refinement of the MFM that dramatically increases UIE performance.

Table 2.2: Performance of Mfa on UIE.

	Random Baseline	MF Alone	Mfa
Accuracy	25%	53%	52%
Labeled Example Equivalent	0	8	8

2.5.5 Analysis of the Performance of Mfa-ssl

Figure 2.5 shows that Mfa-ssl outperforms NB-EM in document classification, but one important question is whether this performance difference is in fact due to the presence of monotonic features. An alternative hypothesis is that the performance improvements are due instead to the utilization of a smaller hypothesis space in Step 2 in Figure 2.2; for smaller training sets, could the regularization inherent in performing feature selection improve generalization accuracy *without* relying on MFs?

We tested this hypothesis by replacing Mfa-ssl's monotonic feature measure $f(y, x_i)$ with a standard information gain measure, and learning an equal number of features distinct from the features selected by Mfa-ssl. We found that this information-gain based approach results in performance essentially equivalent to that of NB-EM, suggesting that a smaller

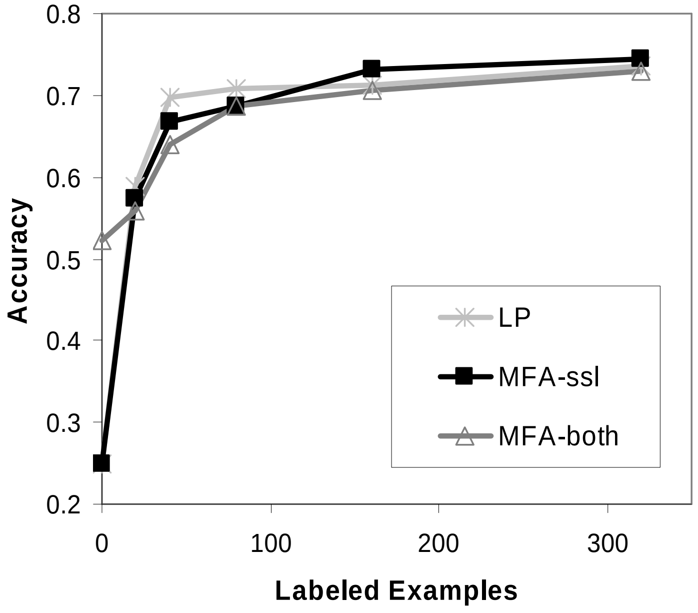

Figure 2.6: Performance in information extraction. Mfa and Mfa-ssl offer similar performance to the baseline label propagation algorithm.

hypothesis space does *not* account for the performance boost due to the monotonic features.

Another question is whether restricting the number of learned monotonic features to a fixed k perhaps biases Mfa-ssl toward a more uniform distribution, which happens to be correct for the 20 newsgroups data set. Indeed, performance of NB-EM can be substantially improved by biasing the algorithm toward a uniform marginal distribution $P(y)$ or by balancing the training sets. However, the individual bootstrapped examples produced in Step 1 of Mfa-ssl have strongly non-uniform distributions, and as mentioned above, selecting in Mfa-ssl a maximum of k features by a criterion other than monotonicity did not improve performance over NB-EM. Lastly, while Mfa-ssl was not effective in the unbalanced extraction experiments in Section 2.5.4, we later show it can be somewhat effective when coupled with the Urns model (see Section 3.4.5). Ultimately, experiments on additional unbalanced data sets are necessary.

2.6 Conclusions

We have shown that even in the complete absence of labeled data, accurate classification is possible given the identity of a feature that varies monotonically with class probability. We have shown that the Monotonic Feature Model (MFM) occurs in several different text-processing applications in practice. We presented a general framework for utilizing Monotonic Features to perform classification without hand-labeled data, or in a semi-supervised setting where monotonic features can be discovered from small numbers of hand-labeled examples. We complemented our experimental results with a theoretical analysis, proving formally that MFs can guarantee PAC learning under certain assumptions, that MFs can be preferable to labeled examples as the dimensionality of the feature space increases, and that the MFM is distinct from problem structures considered previously in semi-supervised learning.

We demonstrated experimentally that the MFM improves accuracy in document classification, in both the semi-supervised and unsupervised settings. However, the model was found to be ineffective for extraction. The following chapter presents techniques that address this limitation of the MFM.

Chapter 3
THE URNS MODEL OF REDUNDANCY IN TEXT

The previous chapter described how Monotonic Features (MFs) can reduce the amount of human labeling required to perform semi-supervised classification. In fact, in some cases MFs can enable classification without *any* labeled examples. However, the MF model (MFM) has important limitations: in many classification settings, merely knowing that a feature is monotonic is *not* sufficient to build an effective classifier. We begin this chapter by demonstrating the limitations of the MFM theoretically.

Our solution to the limitations of the MFM is the Urns model, a special case of the MFM that captures a stronger distributional structure often found in MFs in practice. Specifically, many MFs (including those employed in Chapter 2) are *counts* of observations that each suggest an example is in the target class. Urns models these observations as draws from a mixture of two Zipfian distributions, one for target class elements and another for errors. In general, this is a much stronger distributional assumption than simple monotonicity. Theoretical results show that the Urns model can enable accurate classification in cases in which the MFM alone cannot. Further, in experiments the Urns model is shown to provide much more accurate probability estimates for UIE when compared with using only the MFM, or with other models from previous work.

The contributions of this chapter are as follows:

1. A formal model that, unlike previous work, explicitly models the impact of sample size, redundancy, and different MFs on the probability that an example is a member of the target class.

2. Methods for estimating the model's parameters in the unsupervised, semi-supervised and supervised cases.

3. Experiments in Information Extraction that demonstrate the model's improved per-

formance over the techniques used in previous work to assess the probability that extracted information is correct. For UIE, our model is a factor of 15 closer to the correct log likelihood than the noisy-or model used in previous work; the model is 20 times closer than KnowItAll's Pointwise Mutual Information (PMI) method [23], which is based on Turney's PMI-IR algorithm [57]. For supervised IE, our model achieves a 19% improvement in average log likelihood over the noisy-or model, but is only marginally better than SVMs and logistic regression.

4. Theoretical results that:

   (a) Characterize how target class probability increases with count MF values, as well as how the total number of observations impacts the accuracy of the model.

   (b) Prove that given enough observations, the parameters of the model can be learned from unlabeled data alone—and that the model can overcome the limitations of the MFM for unsupervised classification.

The remainder of the chapter is organized as follows. Section 3.1 establishes theoretically the limitations of the MFM. Section 3.2 introduces our abstract probabilistic model, and Section 3.3 describes its implementation in practice. Section 3.4 reports on experimental results in UIE, and Section 3.5 describes experiments in document classification. We summarize additional ways in which Urns has been employed (beyond IE and document classification) in Section 3.6. Our theoretical results are presented in Section 3.7. Section 3.8 contrasts our model with previous work; the chapter concludes with a discussion of future work.

3.1 Limitations of the MFM

Corollary 4.1 states that the MFM can provide accurate classification even without labeled examples, but relies on strong assumptions. In this section, we illustrate theoretically that these assumptions preclude that the MF values apply to multiple distinct concepts, a situation which is prevalent in practice. This has significant ramifications for the applicability of the monotonic feature approach, and helps explain the relative ineffectiveness of Mfa in IE and UIE demonstrated in Section 2.5.4.

In Corollary 4.1, the classifier constructed from the MF can set its classification threshold at any value, provided that non-zero probability mass lies on either side of the value. While such a classifier is sufficient given the assumptions of the theory, it is not at all guaranteed to be effective in practice, where those assumptions are violated.

The primary difficulty confronted in practice is that MFs for one class are almost always MFs for other, related classes as well. Consider, for example, the list of extractions shown in Table 3.1. Listed are the top ten most frequent extractions for the `City` class in our experiment, along with the number of times each was extracted. While these most frequent extractions are correct instances of `City`, they are also consistent with a multitude of other concepts—for example, they are all cities in the Western Hemisphere, and all cities with opera houses. Likewise, all ten extractions are more generally entities that have governments, and are all located on Earth. If our MF-based classifier uses a threshold such that only these ten examples are positive, it is in general impossible to determine whether the target concept is `City` or another concept.

Table 3.1: The ten most frequently extracted `City` extractions.

Extraction	Frequency
New York	1488
Chicago	999
Los Angeles	859
London	712
San Francisco	682
Boston	676
Paris	454
Philadelphia	440
Washington	395
Seattle	381

More generally, it is often the case due to nested subconcepts like those discussed above that a given MF for one class is also an MF for another distinct but related class. This is problematic for the MFM, because given only the MF as input, it is impossible to tell which class is the target. We formalize this below.

Proposition 11 *If $\mathcal{C}_{\neg M}$ is learnable in the PAC model with classification noise, and concepts $c, c' \in \mathcal{C}$ are such that $P(c(x) = c'(x))$ is bounded away from zero, and each of a set of monotonic features M for the class $c(\mathbf{x}) = 1$ are also monotonic features for the class $c'(\mathbf{x}) = 1$, then $\mathcal{C}$ is not in general learnable from unlabeled data given only M (i.e., the conditional independence assumption is violated).*

Proof. Assume instead that an algorithm exists that can PAC-learn the target concept from only unlabeled data M. For either concept c or c', the input to the algorithm is the same, and hence the outputted classifiers will be the same. If we choose $\delta < P(c(x) = c'(x))/2$, the error rate for at least one of the classifiers learned for c or c' must be larger than δ, a contradiction.$\square$

In practice, we expect MFs that apply to multiple, similar but distinct classes to be ubiquitous in text classification; thus, the above result is discouraging. Fortunately, however, given an MF with a distributional structure based on the Urns model, we can in fact establish PAC learnability from unlabeled data even when the MF may apply to multiple concepts and the conditional independence assumption is violated. We demonstrate this formally in Section 3.7.4.

3.2 The Urns Model

Urns is a model for generating the values of MFs. Our model takes the form of a classic "balls-and-urns" model from combinatorics. We first consider the single urn case, for simplicity, and later generalize to the full multiple *Urns Model* used in our experiments.

The model applies specifically to MFs which represent the *number of occurrences* of some event suggesting an example is a member of a class. For clarity we will refer these MFs as *count MFs.* Count MFs share all the properties of MFs defined in Section 2.3, but are further restricted to take non-negative integral values. As an example, Chapter 2 showed that in document classification, the number of times the class name occurs in a document is an MF; this feature is also a count MF. Similarly, the number of science fiction movies that a person rents could be a count MF for the "Star Wars fan" class. Not *all* MFs are count MFs, however—FICO score, for example, may be an MF for the class of individuals who will repay a loan, but it is not a count MF as it does not represent a count of repeated observations.

The single-urn model for generating count MFs consists of a set of balls in an urn, where each ball has a label denoting the identity of a particular example. Labels for different examples can be repeated on differing numbers of balls in the urn. In the model, values of count MFs are generated by making n repeated draws from the urn (with replacement) and recording how frequently each label is obtained. The MF value for a given example is then simply the number of times its label is drawn from the urn.

As a concrete example, consider the Information Extraction task of classifying strings as members of the `City` class, using a count MF of the number of occurrences of the phrase "cities such as x" in a corpus. The urn for this count MF contains balls labeled with different potential extractions (e.g., "Yakima," "Los Angeles," "California"). Frequently-mentioned cities (e.g. "Paris") are repeated on many balls, whereas more obscure cities (and most errors) appear on fewer balls. If we observe the two phrases "scenic cities such as Yakima" and "Washington cities such as Yakima," this corresponds to the ball labeled "Yakima" being drawn twice from the urn.

Formally, the parameters that characterize an urn are:

- C – the set of unique target labels; $|C|$ is the number of unique target labels in the urn.

- E – the set of unique error labels; $|E|$ is the number of unique error labels in the urn.

- $num(b)$ – the function giving the number of balls labeled by b where $b \in C \cup E$. $num(B)$ is the multi-set giving the number of balls for each label $b \in B$.

Of course, classification systems do not have access to these parameters directly. The goal of a classification system is to discern which of the labels it extracts are in fact elements of C, based on the number of repetitions of each label. Thus, the central question we are investigating is: *given that a particular label x was extracted k times in a set of n draws from the urn, what is the probability that $x \in C$?*

In deriving this probability formally below, we assume the system has access to multi-sets $num(C)$ and $num(E)$ giving the number of times the labels in C and E appear on balls in the urn. In our experiments, we provide methods that estimate these multi-sets in the unsupervised, semi-supervised, and supervised settings. We derive the probability that an element extracted k of n times is of the target class as follows:

First, we have that

$$P(x \,\mathrm{appears}\, k \,\mathrm{times\, in}\, n \,\mathrm{draws}|x \in C) = \sum_r \binom{n}{k} \left(\frac{r}{s}\right)^k \left(1 - \frac{r}{s}\right)^{n-k} P(num(x) = r|x \in C)$$

where s is the total number of balls in the urn, and the sum is taken over possible repetition rates r.

Then we can express the desired quantity using Bayes Rule:

$$P(x \in C|x \,\mathrm{appears}\, k \,\mathrm{times\, in}\, n \,\mathrm{draws}) = \frac{P(x \,\mathrm{appears}\, k \,\mathrm{times\, in}\, n \,\mathrm{draws}|x \in C)P(x \in C)}{P(x \,\mathrm{appears}\, k \,\mathrm{times\, in}\, n \,\mathrm{draws})} \tag{3.1}$$

Note that these expressions include prior information about the label x – for example, $P(x \in C)$ is the prior probability that the string x is a target label, and $P(num(x) = r|x \in C)$ represents the probability that a target label x is repeated on r balls in the urn. In general, integrating this prior information could be valuable for classification systems; however, in the analysis and experiments that follow, we make the simplifying assumption of uniform priors, yielding the following simplified form:

Proposition 12

$$P(x \in C|x \,\mathrm{appears}\, k \,\mathrm{times\, in}\, n \,\mathrm{draws}) = \frac{\sum_{r \in num(C)} (\frac{r}{s})^k (1 - \frac{r}{s})^{n-k}}{\sum_{r' \in num(C \cup E)} (\frac{r'}{s})^k (1 - \frac{r'}{s})^{n-k}}$$

3.2.1 The Uniform Special Case

For illustration, consider the simple case in which all labels from C are repeated on the same number of balls. That is, $num(c_i) = R_C$ for all $c_i \in C$, and assume also that $num(e_i) = R_E$ for all $e_i \in E$. While these assumptions are unrealistic (in fact, we use a Zipf distribution for $num(b)$ in our experiments), they are a reasonable approximation for the majority of labels, which lie on the flat tail of the Zipf curve.

Define p to be the precision of the MF generation process; that is, the probability that a given draw comes from the target class. In the uniform case, we have:

$$p_{USC} = \frac{|C|R_C}{|E|R_E + |C|R_C}$$

The probability that a *particular* element of C appears in a given draw is then $p_C = p_{USC}/|C|$, and similarly $p_E = (1 - p_{USC})/|E|$.

Using a Poisson model to approximate the binomial from Proposition 12, we have:

$$P_{USC}(x \in C | x \, \text{appears} \, k \, \text{times in} \, n \, \text{draws}) \approx \frac{1}{1 + \frac{|E|}{|C|}(\frac{p_E}{p_C})^k e^{n(p_C - p_E)}} \quad (3.2)$$

In order for k to be a monotonic feature, it must be that $p_C > p_E$. Notice that when this is true, Equation (3.2) shows that the odds that $x \in C$ increase exponentially with the number of times k that x is extracted, but also decrease exponentially with the sample size n.

A few numerical examples illustrate the behavior of this equation. The examples assume that the precision p is 0.9. Let $|C| = |E| = 2,000$. This means that $R_C = 9 \times R_E$—target balls are nine times as common in the urn as error balls. Now, for $k = 3$ and $n = 10,000$ we have $P(x \in C) = 93.0\%$. Thus, we see that a small number of repetitions can yield high confidence in an example. However, when the sample size increases so that $n = 20,000$, and the other parameters are unchanged, then $P(x \in C)$ drops to 19.6%. On the other hand, if C balls repeat much more frequently than E balls, say $R_C = 90 \times R_E$ (with $|E|$ set to 20,000, so that p remains unchanged), then $P(x \in C)$ rises to 99.9%.

The above examples enable us to illustrate the advantages of URNS over the noisy-or model used in previous IE work [35, 3]. The noisy-or model for IE assumes that each extraction is an independent assertion, correct a fraction p of the time, that the extracted label is "true." The noisy-or model assigns the following probability to extractions:

$$P_{noisy-or}(x \in C | x \, \text{appears} \, k \, \text{times}) = 1 - (1-p)^k$$

Therefore, the noisy-or model will assign the same probability— 99.9%—in *all three* of the above examples. Yet, as explained above, 99.9% is only correct in the case for which $n = 10,000$ and $R_C = 90 \times R_E$. As the other two examples show, for different sample sizes or repetition rates, the noisy-or model can be highly inaccurate. This is not surprising given that the noisy-or model ignores the sample size and the repetition rates. Section 3.4 quantifies the improvements over the noisy-or obtained by Urns in practice.

3.2.2 Applicability of the Urns *model*

Under what conditions does our redundancy model provide accurate probability estimates? We address this question formally in Section 3.7, but informally two primary criteria must hold. First, labels from the target set C must be repeated on more balls in the urn than labels from the E set, as in Figure 3.1. The shaded region in Figure 3.1 represents the "confusion region" – if we employ only the monotonic feature, half of the examples in this region will be classified incorrectly, even with the ideal classifier and infinite data, because for these examples there simply isn't enough information to decide whether they belong to C or E. Thus, our model is effective when the confusion region is relatively small. Secondly, even for a small confusion region, the sample size n must be large enough to approximate the two distributions shown in Figure 3.1; otherwise the probabilities output by the model will be inaccurate.

3.2.3 Multiple Urns

We now generalize our model to encompass multiple urns. When we have multiple count MFs for the same target class, we could simply sum the count MF values for each example and apply the single-urn model as described in the previous section. However, this approach forfeits differences between the count MFs that may be valuable for classification.

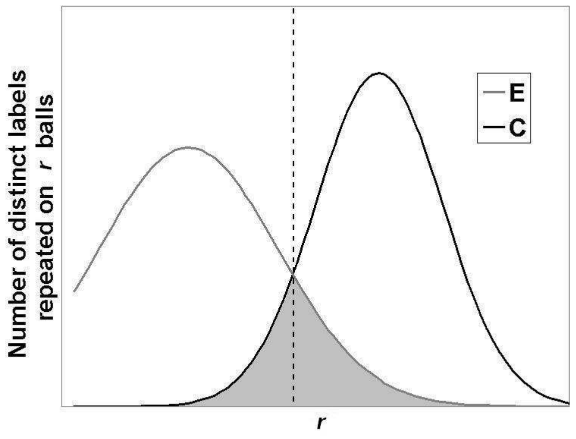

Figure 3.1: Schematic illustration of the number of distinct labels in the C and E sets with repetition rate r. The "confusion region" is shaded.

In IE, for example, information is often extracted using multiple, distinct mechanisms – for example, an IE system might employ several patterns for extracting city names, e.g. "cities including x" and "x and other towns." It is often the case that different patterns have different modes of failure, so extractions appearing across multiple patterns are generally more likely to be true than those appearing for a single pattern. We can model this situation by introducing multiple urns where each urn represents a different pattern.[1]

Instead of n total extractions, in the multi-urn case we have a sample size n_m for each urn $m \in M$, with the label for example x appearing k_m times. Let $A(x, (k_1, \ldots, k_m), (n_1, \ldots, n_m))$ denote this event. Further, let $A_m(x, k, n)$ be the event that label x appears k times in n draws from urn m, and assuming that the draws from each urn are independent, we have:

Proposition 13

$$P(x \in C | A(x, (k_1, \ldots, k_m), (n_1, \ldots, n_m))) = \frac{\sum_{c_i \in C} \prod_{m \in M} P(A_m(c_i, k_m, n_m))}{\sum_{x \in C \cup E} \prod_{m \in M} P(A_m(x, k_m, n_m))}$$

[1]We may lump several patterns into a single urn if they tend to behave similarly.

With multiple urns, the distributions of labels among balls in the urns are represented by multi-sets $num_m(C)$ and $num_m(E)$. Expressing the correlation between $num_m(x)$ and $num_{m'}(x)$ is an important modeling decision. Multiple urns are especially beneficial when the repetition rates for elements of C are more strongly correlated across different urns than they are for elements of E—that is, when $num_m(x)$ and $num_{m'}(x)$ tend to be closer to each other for $x \in C$ than for $x \in E$. Fortunately, this turns out to be the case in practice in IE. We describe our method for modeling multi-urn correlation below.

To model multiple urns, we consider different precisions p_m for each urn, but make the simplifying assumption that the size and shape parameters are the same for all urns. As mentioned in section 3.2, we expect repetition rate correlation across urns to be higher for elements of the C set than for the E set. We model this correlation as follows: first, elements of the C set are assumed to come from the same location on the Zipf curve for all urns, that is, their relative frequencies are perfectly correlated. Some elements of the E set are similar, and have the same relative frequency across urns – these are the *systematic* errors. However, the rest of the E set is made up of *non-systematic* errors, meaning that they appear for only one kind of mechanism (for example, "Eastman Kodak" is extracted as an instance of `Film` only in phrases involving the word "film", and not in those involving the word "movie."). Formally, non-systematic errors are labels that are present in some urns and not in others. Each type of non-systematic error makes up some fraction of the E set, and these fractions are the parameters of our correlation model. Assuming this simple correlation model and identical size and shape parameters across urns is too restrictive in general— differences between mechanisms are often more complex. However, our assumptions allow us to compute probabilities efficiently (as described below) and do not appear to hurt performance significantly in practice.

With this correlation model, if a label x is an element of C or a systematic error, it will be present in all urns. In terms of Proposition 13, the probability that a label x appears k_m times in n_m draws from m is:

$$P(A_m(x, k_m, n_m)) = \binom{n_m}{k_m}(f_m(x))^{k_m}(1 - f_m(x))^{n_m - k_m} \tag{3.3}$$

where $f_m(x)$ is the frequency of label x. That is,

$$\begin{aligned} f_m(c_i) &= p_m Q_C i^{-z_C} \text{ for } c_i \in C \\ f_m(e_i) &= (1-p_m) Q_E i^{-z_E} \text{ for } e_i \in E \end{aligned}$$

In these expressions, i is the frequency rank of the label, assumed to be the same across all urns, and Q_C and Q_E are normalizing constants such that

$$\sum_{c_i \in C} Q_C i^{-z_C} = \sum_{e_i \in E} Q_E i^{-z_E} = 1$$

For a non-systematic error x which is not present in urn m, $P(A_m(x, k_m, n_m))$ is 1 if $k_m = 0$ and 0 otherwise. Substituting these expressions for $P(A_m(x, k_m, n_m))$ into Proposition 13 gives the final form of our Urns model.

3.3 Implementation of the Urns Model

In order to compute probabilities for extractions, we need a method for estimating $num(C)$ and $num(E)$. For the purpose of estimating these sets from tagged or untagged data, we assume that $num(C)$ and $num(E)$ are Zipf distributed, meaning that if c_i is the ith most frequently repeated label in C, then $num(c_i)$ is proportional to i^{-z_C}. We can then characterize the $num(C)$ and $num(E)$ sets with five parameters: the set sizes $|C|$ and $|E|$, the shape parameters z_C and z_E, and the extraction precision p.

3.3.1 Efficient Computation

A feature of our implementation is that it allows for efficient computation of probabilities. In general, computing the sum in Proposition 13 over the potentially large C and E sets would require significant computation for each example. However, given a fixed number of urns, with $num(C)$ and $num(E)$ Zipf distributed, an integral approximation to the sum in Proposition 13 (using a Poisson in place of the binomial in Equation 3.3) can be solved in closed form in terms of incomplete Gamma functions. The details of this approximation and its solution for the single-urn case are given in Section 3.7.[2] The closed form expression

[2]For the multi-urn solution, which is obtained through a symbolic integration package and therefore complicated, we refer the reader to the Java implementation of the solution which is available for download – see Appendix A.

can be evaluated quickly, and thus probabilities for examples can be obtained efficiently. This solution leverages our assumptions that size and shape parameters are identical across urns, and that relative frequencies are perfectly correlated. Finding efficient techniques for computing probabilities under less stringent assumptions is an item of future work.

3.3.2 Supervised Parameter Estimation

In the event that a large sample of hand-tagged training examples is available for each target class of interest, we can directly estimate each of the parameters of Urns. In our experiments, we use a population-based stochastic optimization technique to identify parameter settings that maximize the conditional log likelihood of the training data.[3] Once the parameters are set, the model yields a probability for each extraction, given the number of times k_m it appears in each urn and the number of draws n_m from each urn.

This section describes how we implement Urns for both UIE and supervised IE, and identifies the assumptions made in each case.

3.3.3 Unsupervised Parameter Estimation

Estimating model parameters in an unsupervised setting requires making a number of assumptions tailored to the specific task. Below, we first detail the assumptions employed in Urns for UIE. It is important to note that while these assumptions are specific to UIE, they are *not* specific to a particular target class. As argued in [24], UIE systems cannot rely on per-class information—in the form of either assumptions or hand-tagged training examples—if they are to scale to extracting information on arbitrary classes that are not specified in advance. We then present the set of assumptions employed in Urns for document classification.

Unsupervised Parameter Estimation for UIE

Implementing Urns for UIE requires a solution to the challenging problem of estimating $num(C)$ and $num(E)$ using only untagged data. Let U be the multi-set consisting of the

[3]Specifically, we use the Differential Evolution routine built into Mathematica 5.0.

number of times each unique label was extracted in a given corpus. $|U|$ is the number of unique labels encountered (i.e. the number of unlabeled examples), and the sample size $n = \sum_{r \in U} r$.

In order to learn $num(C)$ and $num(E)$ from untagged data, we make the following assumptions:

- Because the number of different possible errors is nearly unbounded, we assume that the error set is very large.[4]

- We assume that both $num(C)$ and $num(E)$ are Zipf distributed where the z_E parameter is set to 1.

- In our experience with KnowItAll, we found that while different extraction rules have differing precision, each rule's precision is stable across different classes [24]. Urns takes this precision as an input. To demonstrate that Urns is not overly sensitive to this parameter, we chose a fixed value (0.9) and used it as the precision p_m for all urns in our experiments. [5] Section 3.4.4 provides evidence that the observed p value tends to be relatively stable across different target classes.

We then use Expectation Maximization (EM) over U in order to arrive at appropriate values for $|C|$ and z_C (these two quantities uniquely determine $num(C)$ given our assumptions). Our EM algorithm proceeds as follows:

1. Initialize $|C|$ and z_C to starting values.

2. Repeat until convergence:

   (a) **E-step** Assign probabilities to each element of U using Proposition (12).

[4]In our experiments, we set $|E| = 10^6$. A sensitivity analysis showed that changing $|E|$ by an order of magnitude, in either direction, resulted in only small changes to our results.

[5]A sensitivity analysis showed that choosing a substantially higher (0.95) or lower (0.80) value for p_m still resulted in Urns outperforming the noisy-or model by at least a factor of 8 and PMI by at least a factor of 10 in the experiments described in section 3.4.1.

(b) **M-step** Set $|C|$ and z_C from U using the probabilities assigned in the E-step (details below).

We obtain $|C|$ and z_C in the M-step by first estimating the rank-frequency distribution for labels from C in the untagged data. From the untagged data and the probabilities found in the E-step, we can obtain $E_C[k]$, the expected number of labels from C that were extracted k times. We then round these fractional expected counts into a discrete rank-frequency distribution with a number of elements equal to the expected total number of labels from C in the untagged data, $\sum_k E_C[k]$. We obtain z_C by fitting a Zipf curve to this rank-frequency distribution by linear regression on a log-log scale.[6]

Lastly, we set $|C| = \sum_k E_C[k] + \mathit{unseen}$, where we estimate the number of unseen labels of the C set using Good-Turing estimation ([29]). Good-Turing estimation provides an estimate of the *probability mass* of the unseen labels (specifically, the estimate is equal to the expected fraction of the draws from C that extracted labels seen only once). We simply assume that each unseen element has probability equal $\sum_k E_C[k]^{-z_C}$. A more accurate method might be to choose *unseen* such that the actual number of unique elements observed equals that expected by the model (where the latter is measured e.g. by sampling). Such methods are an item of future work.

This unsupervised learning strategy proved effective for target relations of different sizes; for example, the number of elements of the `Country` relation with non-negligible extraction probability was about two orders of magnitude smaller than that of the `Film` and `City` relations.

Semi-supervised Parameter Estimation

When both unlabeled and labeled data are available, as in MFA-SSL and MFA-BOTH, we use a single-urn model. The precision p of the urn is estimated from the labeled data (using the f function defined in Section 2.5.1). We perform the same EM-style parameter estimation over the unlabeled data as detailed for UIE, setting the multi-set U to the collection of MF

[6]To help ensure that our probability estimates are increasing with k, if z_C falls below 1, we adjust z_E to be less than z_C.

values in D_U, with one important change. Instead of assuming $|E| = 10^6$, because the total number of unlabeled examples u is known, we employ the constraint that $|C| + |E| = u$.

Unsupervised Parameter Estimation for Document Classification

The assumptions we employ in Urns for document classification are similar to the single-urn assumptions employed for UIE. The differences are that we employ the constraint $|C|+|E| = u$ as above and, because the count MFs are less precise, we employ a p value of 0.5.

3.4 Experimental Results in IE

How accurate is Urns at assigning probabilities of correctness to extractions? And can these probabilities improve on the limitations of the MFM detailed in Section 3.1? In this section, we answer these questions by comparing the accuracy of Urns's probabilities against other methods from previous work, and evaluating how Urns performs when used as a component of Mfa.

This section begins by describes our experimental results for IE under two settings: unsupervised and supervised. Semi-supervised experiments using Urns with Mfa are later described in Section 3.4.5. We begin by describing the two unsupervised methods used in previous work: the noisy-or model and PMI. We then compare Urns with these methods experimentally, and lastly compare Urns with several baseline methods in a supervised setting.

We evaluated our algorithms on extraction sets for the relations `City(x)`, `Film(x)`, `Country(x)`, and `MayorOf(x,y)`, taken from experiments performed in [24]. The sample size n was 64,605 for `City`, 135,213 for `Film`, 51,390 for `Country` and 46,858 for `MayorOf`. The extraction patterns were partitioned into urns based on the name they employed for their target relation (e.g. "country" or "nation") and whether they were left-handed (e.g. "countries including x") or right-handed (e.g. "x and other countries"). Each combination of relation name and handedness was treated as a separate urn, resulting in four urns for each of `City(x)`, `Film(x)`, and `Country(x)`, and two urns for `MayorOf(x)`.[7] [8]

For each relation, we tagged a sample of 1000 extracted labels, using external knowledge bases (the Tipster Gazetteer for cities and the Internet Movie Database for films) and manually tagging those instances not found in a knowledge base. In the UIE experiments, we evaluate our algorithms on all 1000 examples, and in the supervised IE experiments we perform 10-fold cross validation.

3.4.1 UIE Experiments

We compare Urns against two other methods for unsupervised information extraction. First, in the *noisy-or* model used in previous work, an extraction appearing k times is assigned probability $1 - \prod_{m \in M}(1 - p_m)^k$, where p_m is the extraction precision for urn m. We describe the second method below.

Pointwise Mutual Information

Our previous work on KnowItAll used Pointwise Mutual Information (PMI) to obtain probability estimates for extractions [24]. Specifically, the PMI between an extraction and a set of automatically generated *discriminator phrases* (e.g., "movies such as x") is computed from Web search engine hit counts. These PMI scores are used as features in a Naive Bayes Classifier (NBC) to produce a probability estimate for the extraction. The NBC is trained using a set of automatically bootstrapped seed instances. The positive seed instances are taken to be those having the highest PMI with the discriminator phrases after the bootstrapping process; the negative seeds are taken from the positive seeds of other

[7]Draws from Urns are intended to represent independent extractions. Because the same sentence can be duplicated across multiple different Web documents, in these experiments we consider only each *unique* sentence containing an extraction to be a draw from Urns. In experiments with other possibilities, including counting the number of unique documents producing each extraction, or simply counting every occurrence of each extraction, we found that performance differences between the various approaches were negligible for our task.

[8]In the unsupervised setting, we assumed that the fraction of errors in the urns that are non-systematic is 0.1, and that errors appearing for only left- or only right-handed rules were equally prevalent to those appearing for only one label. The only exception was the `City` class, where because the target class was the union the the two labels ("city" and "town") rather than the intersection (as with "film" and "movie"), we assumed that no non-systematic errors appeared for only one label. Altering these settings (or indeed, simply using a single urn – see Section 3.4.3) had negligible impact on the results in Figure 3.2.

relations, as in other work (e.g., [35]).

Although PMI was shown in [24] to order extractions fairly well, it has two significant shortcomings. First, obtaining the hit counts needed to compute the PMI scores is expensive, as it requires a large number of queries to web search engines. Second, the seeds produced by the bootstrapping process tend not to be representative of the overall distribution of extractions. This combined with the probability polarization introduced by the NBC tends to give inaccurate probability estimates.

Discussion of UIE Results

The results of our unsupervised experiments are shown in Figure 3.2. We plot deviation from the *ideal* log likelihood—defined as the maximum achievable log likelihood given our feature set.

Our experimental results demonstrate that URNS overcomes the weaknesses of PMI. First, URNS's probabilities are far more accurate than PMI's, achieving a log likelihood that is a factor of 20 closer to the ideal, on average (Figure 3.2). Second, URNS is substantially more efficient as shown in Table 3.2.

This efficiency gain requires some explanation. KNOWITALL relies on queries to Web search engines to identify Web pages containing potential extractions. The number of queries KNOWITALL can issue daily is limited, and querying over the Web is, by far, KNOWITALL's most expensive operation. Thus, number of search engine queries is our efficiency metric. Let d be the number of discriminator phrases used by the PMI method as explained in Section 3.4.1. The PMI method requires $O(d)$ search engine queries to compute the PMI of each extraction from search engine hit counts. In contrast, URNS computes probabilities *directly* from the set of extractions—requiring *no* additional queries, which cuts KNOWITALL's queries by factors ranging from 1.9 to 17.

As explained in section 3.2, the noisy-or model ignores target set size and sample size, which leads it to assign probabilities that are far too high for the `Country` and `MayorOf` relations, where the average number of times each label is extracted is high (see bottom row of Table 3.2). This is further illustrated for the `Country` relation in Figure 3.3. The noisy-or model assigns appropriate probabilities for low sample sizes, because in this case the overall precision of extracted labels is in fact fairly high, as predicted by the noisy-or model. However, as sample size increases relative to the number of true countries, the overall precision of the extracted labels decreases—and the noisy-or estimate worsens. On the other hand, Urns avoids this problem by accounting for the interaction between target set size and sample size, adjusting its probability estimates as sample size increases. Given sufficient sample size, Urns performs close to the ideal log likelihood, improving slightly with more samples as the estimates obtained by the EM process become more accurate. Overall, Urns assigns far more accurate probabilities than the noisy-or model, and its log likelihood is a factor of 15 closer to the ideal, on average. The very large differences between Urns and both the noisy-or model and PMI suggest that, even if the performance of Urns degrades in other domains, it is quite likely to still outperform both PMI and the noisy-or model.

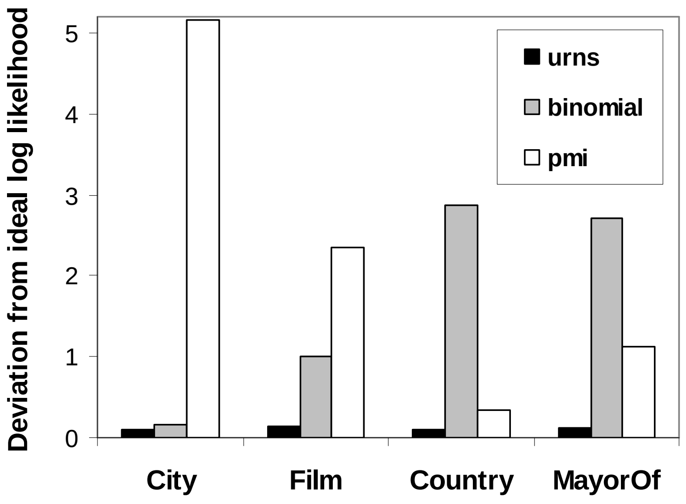

Figure 3.2: Deviation of average log likelihood from the ideal for four relations (lower is better). On average, Urns outperforms noisy-or by a factor of 15, and PMI by a factor of 20.

Our computation of log-likelihood contains a numerical detail that could potentially influence our results. To avoid the possibility of a likelihood of zero, we restrict the probabilities generated by Urns and the other methods to lie within the range (0.00001, 0.99999).

Table 3.2: Improved Efficiency Due to Urns. The top row reports the number of search engine queries made by KnowItAll using PMI divided by the number of queries for KnowItAll using Urns. The bottom row shows that PMI's queries increase with k—the average number of distinct labels for each relation. Thus, speedup tends to vary inversely with the average number of times each label is drawn.

	`City`	`Film`	`MayorOf`	`Country`
Speedup	17.3x	9.5x	1.9x	3.1x
Average k	3.7	4.0	20.7	23.3

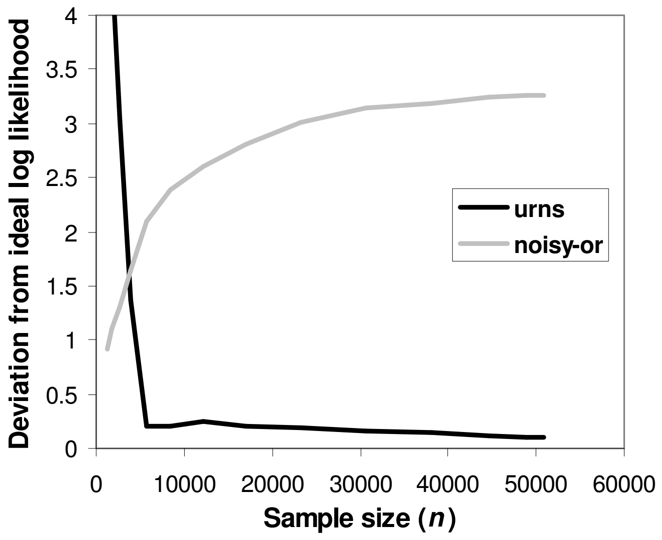

Figure 3.3: Deviation of average log likelihood from the ideal as sample size varies for the `Country` relation (lower is better). Urns performs close to the ideal given sufficient sample size, whereas noisy-or becomes less accurate as sample size increases.

Widening this range tended to improve URNS's performance relative to the other methods, as this increases the penalty for erroneously assigning extreme probabilities—a problem more prevalent for PMI and noisy-or than for URNS. If we narrow the range by two digits of precision, to (0.001, 0.999), URNS still outperforms PMI by a factor of 15, and noisy-or by a factor of 13. Thus, we are comfortable that the differences observed are not an artifact of this design decision.

Lastly, although we focus of our evaluation on the quality of each method's probability estimates in terms of likelihood, the advantage of URNS is also reflected in other metrics such as classification accuracy. When we convert each method's probability estimate into a classification (positive for an example *iff* the probability estimate is greater than 0.5), we find that URNS has an average accuracy of approximately 81%, compared with pmi at 63% and noisy-or at 47%. Thus, URNS decreases classification error over the previous methods by a factor of 1.9x to 2.8x.

3.4.2 Supervised IE Experiments

We compare URNS with three supervised methods. All methods utilize the same feature set as URNS, namely the extraction counts k_m.

- **noisy-or** – Has one parameter per urn, making a set of M parameters $(h_1, \ldots, h_M)$, and assigns probability equal to

$$1 - \prod_{m \in M} (1 - h_m)^{k_m}.$$

- **logistic regression** – Has $M+1$ parameters $(a, b_1, b_2, \ldots, b_M)$, and assigns probability equal to

$$\frac{1}{1 + e^{a + \sum_{m \in M} k_m b_m}}.$$

- **SVM** – Consists of an SVM classifier with a Gaussian kernel. To transform the output of the classifier into a probability, we use the probability estimation built-in to LIBSVM [10], which is based on logistic regression of the SVM decision values.

Parameters maximizing the conditional likelihood of the training data were found for the noisy-or and logistic regression models using Differential Evolution. For those models and Urns, we performed 20 iterations of Differential Evolution using 400 distinct search points. In the SVM case, we performed grid search to find the kernel parameters giving the best likelihood performance for each training set – this grid search was required to get acceptable performance from the SVM on our task.

The results of our supervised learning experiments are shown in Table 3.3. Urns, because it is more expressive, is able to outperform the noisy-or and logistic regression models. In terms of deviation from the ideal log likelihood, we find that on average Urns outperforms the noisy-or model by 19%, logistic regression by 10%, but SVM by only 0.4%.

Table 3.3: Supervised IE experiments. Deviation from the ideal log likelihood for each method and each relation (lower is better). The overall performance differences are small, with Urns 19% closer to the ideal than noisy-or, on average, and 10% closer than logistic regression. The overall performance of SVM is close to that of Urns.

	City	Film	Mayor	Country	Average
noisy-or	0.0439	0.1256	0.0857	0.0795	0.0837
logistic regression	0.0466	0.0893	**0.0655**	0.1020	0.0759
SVM	0.0444	0.0865	0.0659	**0.0769**	0.0684
Urns	**0.0418**	**0.0764**	0.0721	0.0823	**0.0681**

3.4.3 Benefit from Multiple Urns

The previous results use the full multi-urn model. How much of Urns's large performance advantage in UIE is due to multiple urns?

In terms of likelihood, as measured in Figure 3.2, we found that the impact of multiple urns is negligible. This is primarily because the majority of extractions occur only a handful of times, and for these extractions the multiple-urn model lacks enough data to estimate the correlation of counts across urns. Multiple urns can offer some performance benefit, however, for more common extractions.

We evaluated the effect of multiple urns across the four relations shown in Figure 3.2. The average precision of the top K extractions, ranking by either the single-urn or full URNS model in decreasing order of probability, is shown in Table 3.4 for varying K. The full URNS model always performs at least as well as the single-urn model, and sometimes provides much more precise extractions. In fact, using multiple urns reduces the error by 29% on average for the five K values shown in the table.

Table 3.4: Precision of the K highest-ranked extractions for varying values of K between 10 and 200. Across the five K values shown, URNS reduces error over the single-urn model by an average of 29%.

Number of highest-ranked extractions	Single Urn	URNS
10	1	1
20	0.9875	1
50	0.925	0.955
100	0.8375	0.845
200	0.7075	0.71

3.4.4 Is p a "universal constant"?

Our experiments employed an extraction precision parameter p of 0.9. While URNS still massively outperforms previous methods even if this value is adjusted to 0.8 or 0.95, the accuracy of URNS's probabilities does degrade as p is altered away from 0.9.

In this section, we attempt to measure how consistent the observed p value is across varying classes. This experiment differs somewhat from those presented above. In order to test across a wide variety of classes, we moved beyond the experiments from [24] and used the TEXTRUNNER system to provide instances of classes [5]. To choose classes to investigate, we randomly selected 12 nouns from WordNet for which there were at least 100 extractions in TEXTRUNNER. We excluded nouns which were overly general such that nearly any extraction would be correct (e.g., the class `Example`) and nouns which are rarely or never used to name concrete instances (e.g., the class `Purchases`). The results in this section were compiled by querying TextRunner for 100 sentences containing extractions of each class. [9]

While TextRunner provides greater coverage than KnowItAll, precision in general is lower. One of the inaccuracies of the TextRunner system is that it often fails to delimit the boundaries of extractions properly (e.g., it extracts the phrase "alkanes or cycloalkanes" as an instance of the `Solvents` class). We found that we could improve the precision of TextRunner by over 20% on average by post-processing all extractions, breaking on conjunctions or punctuation (i.e. the previous example becomes simply "alkanes"). Our results employ this heuristic.

The results of the experiment are shown in Table 3.5. For each class, "p Observed" gives the fraction of the 100 extractions tagged correct (by manual inspection). The average p value observed across classes of 0.84 is lower than the value of 0.9 we use in our experiments; this reflects the relatively lower precision of TextRunner as well as the increased difficulty of extracting common nouns (versus the proper noun extractions from Section 3.4). The results show that while there is substantial regularity in observed p values, the values are not perfectly consistent. In fact, three classes (with "p Observed" values in bold) differ significantly from the average observed p value (at significance level of 0.01, Fisher Exact Test).

Given that we observe variability in p values across classes, an important question is whether the correct p value can be predicted for a given class. We observed empirically that the precision of extractions for a class increases with how relatively frequently the class name is used in extraction patterns. As an example, the phrase "cultures such as x" appears infrequently relative to the word "cultures," as shown the Table 3.5 in terms of Web hit counts obtained from a search engine. In turn, the class `Cultures` exhibits a relatively low p value. Intuitively, this result makes sense—class names which are more "natural" for naming instances should both appear more frequently in extraction patterns, and provide more precise extractions.

[9]The list of excluded nouns and the labeled extractions for each selected class are available for download; see Appendix A.

We can exploit the above intuition by adjusting the estimate of extraction precision for each class by a factor h_{class}. For illustration, using the values in Table 3.5, we devised the following adjustment factor:

$$h_{class} = 0.08(-2.36 - \log_{10} \frac{\text{Hits(class such as)}}{\text{Hits(class)}}) \tag{3.4}$$

Obviously, this expression is heuristic and could be further refined using additional experiments. However, adjusting by the factor does improve consistency. For example, the quantity "p Observed + h_{class}" has only 57% of the variance of the original "p Observed" (and the same mean, by construction). Further, none of the observed differences of "p Observed - h_{class}" are statistically significantly different from the original mean, using the same significance test employed previously.

Lastly, we should mention that even *without* any adjustment factor, the variance in p value across classes is not substantially greater than that employed in our sensitivity analysis in Section 3.4. Thus, we would expect the performance advantages of Urns over the noisy-or and PMI models to extend to these other classes as well.

3.4.5 Using Urns *in* Mfa

In Section 3.1, we illustrated how the MFM is inadequate for performing accurate classification in certain domains, including UIE. Can Urns be used to overcome these limitations?

We experimented using Urns to assign probabilistic labels to examples to create the sets D'_L in Mfa and Mfa-ssl; we denote the new algorithms as Mfa (Urns) and Mfa-ssl(Urns). The results in UIE, following the experimental setup given in Section 2.5.4, are shown in Table 3.6. Urns improves performance substantially, achieving the equivalent accuracy of the baseline label propagation algorithm trained with over 320 labeled examples. Bootstrapping from the output of Urns using the other features in Mfa (Urns) decreases performance slightly versus Urns alone. Like the similar performance difference between the MF in isolation and Mfa (observed in Section 2.5.4), we expect that the decreased performance is due to the task difficulty and dependencies within the feature set.

Does Urns improve performance in a semi-supervised setting, when learning monotonic features? Figure 3.4 shows that when Mfa-ssl uses urns rather than Naive Bayes to assign

Table 3.5: Average p values for various classes, measured from 100 hand-labeled examples per class. Three of the 12 classes have p values in bold, indicating a statistically significantly difference from the mean of 0.84 (significance level of 0.01, Fisher Exact Test). However, if we adjust the estimate of p per class according to how frequently it occurs in the "such as" pattern (using the factor h_{class}; see text), none of the resulting $p - h_{class}$ values are significantly different from the mean.

`Class`	p Observed	Hits(class such as)/Hits(class)	p Observed + h_{class}
`solvents`	**0.98**	0.201	0.85
`devices`	0.93	0.022	0.87
`thinkers`	0.93	0.013	0.89
`relaxants`	0.92	0.010	0.89
`mushrooms`	0.86	0.001	0.90
`mechanisms`	0.85	0.017	0.80
`resorts`	0.85	0.002	0.88
`flies`	0.84	0.0004	0.93
`tones`	0.77	0.001	0.83
`wounds`	0.77	0.002	0.80
`machines`	**0.69**	0.002	0.71
`cultures`	**0.67**	0.002	0.70

Table 3.6: Performance of MFA and urns on UIE. URNS increases performance over MFA considerably, to 79%, equivalent to the baseline semi-supervised technique (label propagation) trained with 320 labeled examples.

	Random Baseline	MFA	URNS Alone	MFA (URNS)
Accuracy	25%	52%	79%	75%
Labeled Example Equivalent	0	8	>320	>320

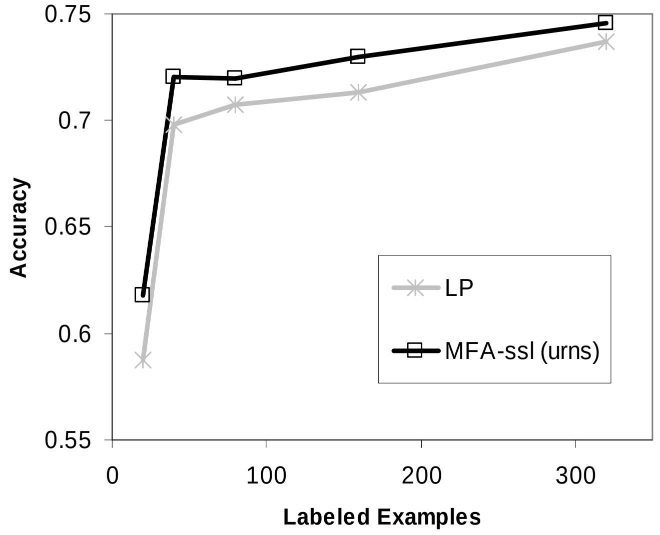

Figure 3.4: Performance of Mfa-ssl augmented with Urns. Employing Urns with Mfa-ssl consistently reduces error, by 6% on average, over the baseline label propagation algorithm (LP).

probabilities based on MFs (Mfa-ssl (urns)), error is consistently reduced over the baseline label propagation algorithm by an average of 6%. This performance is also 7% better on average than Mfa-ssl alone from the experiments in Figure 2.6.

3.5 Experimental Results in Document Classification

We performed experiments in document classification, experimenting with both Mfa (Urns) and Mfa-ssl (Urns). We found the performance differences between these methods and the original Mfa and Mfa-ssl algorithms to be negligible.

We attribute the fact that Urns does not improve performance to two factors. The first is that because the documents in this case are relatively small (newsgroup postings) and the class names are relatively specific, the technique in Mfa of simply assigning documents to a class whenever the class name appears was very effective. The second reason is that in this data set, unlike in extraction, the target set size $|C|$ is the same for all classes; this obviates one of the primary values of Urns, its ability to learn the parameters of $num(C)$ using unlabeled data. Experimenting with more natural document classification data sets that don't obey this restriction is an item of future work.

3.6 Other Applications of Urns

Urns is a general model. In any classification problem, if one of the features signifies a count of observations that has an overall Zipf distribution, the Urns model can be employed. In this section, we highlight three examples of how the Urns model has been applied to tasks other than assigning probabilities of correctness to extractions.

3.6.1 Estimating UIE precision and recall

An attractive feature of Urns is that it enables us to estimate its expected recall and precision as a function of sample size. If the distributions in Figure 3.1 cross at the dotted line shown then, given a sufficiently large sample size n, expected recall will be the fraction of the area under the C curve lying to the right of the dotted line.

For a given sample size n, define τ_n to be the least number of appearances k at which an extraction is more likely to be from the C set than the E set (given the distributions in Figure 3.1, τ_n can be computed using Proposition 12). Then we have:

$$\mathbf{E}[TruePositives] =$$

$$|C| - \sum_{r \in num(C)} \sum_{k=0}^{\tau_n - 1} \binom{n}{k} \left(\frac{r}{s}\right)^k \left(1 - \frac{r}{s}\right)^{n-k}$$

where we define "true positives" to be the number of extracted labels $c_i \in C$ for which the model assigns probability $P(c_i \in C) > 0.5$.

The expected number of false positives is similarly:

$$\mathbf{E}[FalsePositives] =$$

$$|E| - \sum_{r \in num(E)} \sum_{k=0}^{\tau_n - 1} \binom{n}{k} \left(\frac{r}{s}\right)^k \left(1 - \frac{r}{s}\right)^{n-k}$$

The expected precision of the system can then be approximated as:

$$\mathbf{E}[Precision] \approx \frac{\mathbf{E}[TruePositives]}{\mathbf{E}[FalsePositives] + \mathbf{E}[TruePositives]}$$

To illustrate the potential benefit of the above calculations and evaluate their accuracy, we computed expected recall and precision for the particular $num(C)$ and $num(E)$ learned (in the unsupervised setting) in our experiments in Section 3.4. The results appear in Table 3.7. The recall estimates are within 11% of the actual recall for the `City` and `Film` classes. Further, the estimates reflect the important qualitative difference between the large `City` and `Film` classes as compared with the smaller `MayorOf` and `Country` classes.

Were we to increase the sample size n for the `Film` class and the `Country` class each to 1,000,000, the model predicts that we would increase our `Film` recall by 81%, versus only 4% for `Country`. Thus, the above equations allow an information extraction system to dynamically choose how to allocate resources to match given precision and recall goals, even in the absence of hand-tagged data.

Table 3.7: Estimating precision and recall in UIE. Listed is the Urns model estimate for precision and recall, along with the actual measured quantities, for four classes. The major differences between the classes—that the `MayorOf` and `Country` classes have roughly two orders of magnitude lower recall than the `City` and `Film` classes—is qualitatively reflected by the model.

	`City`	`Film`	`Country`	`MayorOf`
n	64605	135213	51390	46858
E[Recall]	12900	37	25900	58
Actual Recall	14300	176	23400	158
E[Precision]	0.78	0.63	0.79	0.62
Actual Precision	0.84	0.77	0.68	0.79

3.6.2 Estimating the functionality of relations

Consider the task of automatically detecting contradictions in text [17]. One fruitful to approach this problem is to identify relations expressed in text that should be *functional*; for example, if we know that the `Headquartered` relation is functional, and we see one document asserting that *Intel* is headquartered in *Santa Clara*, and another asserting it is headquartered in *Phoenix*, we can determine that either one of the documents has an error, or we have made an error in extraction. Leveraging functional relations has been recently shown to be valuable for identifying contradictions in text [51]. Further, manually-detected functional relations have been previously employed to automatically identify extractor errors in IE [3]. Here, we illustrate how URNS can be used to automatically compute the probability that a phrase denotes a function.

The discussion in this section is based on a set of extracted *tuples*. An extracted tuple takes the form $R(x, y)$ where (roughly) x is the subject of a sentence, y is the object, and R is a phrase denoting the relationship between them. If the relation denoted by R is functional, then typically the object y is a function of the subject x. Thus, our discussion focuses on this possibility, though the analysis is easily extended to the symmetric case.

The main evidence that a relation $R(x, y)$ is functional comes from the distribution of y values for a given x value. If R denotes a function and x is unambiguous, then we expect the extractions to be predominantly a single y value, with a few outliers due to noise.

Example A in Figure 3.5 has strong evidence for a functional relation. 66 out of 70 extractions for *was_born_in (Mozart, PLACE)* have the same y value. An ambiguous x argument, however, can make a functional relation appear non-functional. Example B refers to multiple real-world individuals named "John Adams" and has a distribution of y values that appears less functional than example C, which has a non-functional relation.

Logically, a relation R is functional in a variable x if it maps it to a unique variable y: $\forall x, y_1, y_2 R(x, y_1) \wedge R(x, y_2) \Rightarrow y_1 = y_2$. Thus, given a large random sample of ground instances of R, we could detect with high confidence whether R is functional. In text, the situation is far more complex due to ambiguity, polysemy, synonymy, and other linguistic phenomena.

To decide whether R is functional in x for all x, we first consider how to detect whether R is *locally functional* for a particular value of x. We later combine the local functionality probabilities to estimate the global functionality of a relation.[10] Local functionality for a given x can be modeled in terms of the global functionality of R and the ambiguity of x. We later outline an EM-style algorithm that alternately estimates the probability that R is

[10]We compute global functionality as the average local scores, weighted by the probability that x is unambiguous.

A. was_born_in(Mozart, PLACE):

  Salzburg(66), Germany(3), Vienna(1)

B. was_born_in(John Adams, PLACE):

  Braintree(12), Quincy(10), Worcester(8)

C. lived_in(Mozart, PLACE):

  Vienna(20), Prague(13), Salzburg(5)

Figure 3.5: Functional relations such as example A have a different distribution of y values than non-functional relations such as C. Ambiguous x argument as in B, however, can make a functional relation *appear* non-functional.

functional and the probability that x is ambiguous.

Let θ_R^f be the probability that $R(x, \cdot)$ is locally functional for a random x, and let Θ^f be the vector of these parameters across all relations R. Likewise, θ_x^u represents the probability that x is locally unambiguous for random R, and Θ^u the vector for all x.

We wish to determine the *maximum a posteriori* (MAP) functionality and ambiguity parameters given the observed data D, that is $\arg\max_{\Theta^f, \Theta^u} P(\Theta^f, \Theta^u | D)$. By Bayes Rule:

$$P(\Theta^f, \Theta^u | D) \propto P(D | \Theta^f, \Theta^u) P(\Theta^f, \Theta^u) \tag{3.5}$$

We outline a generative model for the data, $P(D|\Theta^f, \Theta^u)$. Let R_x^* indicate the event that the relation R is locally functional for the argument x, and that x is locally unambiguous for R. Also, let D indicate the set of observed tuples, and define $D_{R(x,\cdot)}$ as the multi-set containing the frequencies for extractions of the form $R(x, \cdot)$.

Let us assume that the event R_x^* depends only on θ_R^f and θ_x^u, and further assume that given these two parameters, local ambiguity and local functionality are conditionally independent. We obtain the following expression for the probability of R_x^* given the parameters:

$$P(R_x^* | \Theta^f, \Theta^u) = \theta_R^f \theta_x^u$$

We assume each set of data $D_{R(x,\cdot)}$ is generated independently of all other data and parameters, given R_x^*. From this and the above we have:

$$P(D|\Theta^f, \Theta^u) = \prod_{R,x} \Big(P(D_{R(x,\cdot)}|R_x^*)\theta_R^f \theta_x^u$$
$$+P(D_{R(x,\cdot)}|\neg R_x^*)(1 - \theta_R^f \theta_x^u) \Big) \tag{3.6}$$

These independence assumptions allow us to express $P(D|\Theta^f, \Theta^u)$ in terms of distributions over $D_{R(x,\cdot)}$ given whether or not R_x^* holds. We use a single-urn model to estimate these probabilities based on binomial distributions.

Let $k = \max D_{R(x,\cdot)}$, and let $n = \sum D_{R(x,\cdot)}$; we will approximate the distribution over $D_{R(x,\cdot)}$ in terms of k and n. In the single-urn model, if $R(x,\cdot)$ is locally functional and unambiguous, k has a binomial distribution with parameters n and p, where p is the precision of the extraction process. If $R(x,\cdot)$ is *not* locally functional and unambiguous, then we expect k to typically take on smaller values. Empirically, we find that the underlying frequency of the most frequent element in the $\neg R_x^*$ case tends to follow a Beta distribution.

Under the model, the probability of the evidence given R_x^* is:

$$P(D_{R(x,\cdot)}|R_x^*) \approx P(k,n|R_x^*) = \binom{n}{k} p^k (1-p)^{n-k}$$

And the probability of the evidence given $\neg R_x^*$ is:

$$P(D_{R(x,\cdot)}|\neg R_x^*) \approx P(k,n|\neg R_x^*)$$
$$= \tbinom{n}{k} \int_0^1 \tfrac{p'^{k+\alpha_f-1}(1-p')^{n+\beta_f-1-k}}{B(\alpha_f,\beta_f)} dp'$$
$$= \frac{\binom{n}{k}\Gamma(n-k+\beta_f)\Gamma(\alpha_f+k)}{B(\alpha_f,\beta_f)\Gamma(\alpha_f+\beta_f+n)} \tag{3.7}$$

where n is the sum over $D_{R(x,\cdot)}$, Γ is the Gamma function and B is the Beta function. α_f and β_f are the parameters of the Beta distribution for the $\neg R_x^*$ case (in practice, these are estimated empirically).

Estimating Functionality and Ambiguity

Substituting Equation 3.7 into Equation 3.6 and applying an appropriate prior gives the probability of parameters Θ^f and Θ^u given the observed data D. However, Equation 3.6

contains a large product of sums—with two independent vectors of coefficients, Θ^f and Θ^u—making it difficult to optimize analytically.

If we knew which arguments were ambiguous, we would ignore them in computing the functionality of a relation. Likewise, if we knew which relations were non-functional, we would ignore them in computing the ambiguity of an argument. Instead, we initialize the Θ^f and Θ^u arrays randomly, and then execute an EM-style algorithm to arrive at a high-probability setting of the parameters.

Note that if Θ^u is fixed, we can compute the expected fraction of locally unambiguous arguments x for which R is locally functional, using $D_{R(x',\cdot)}$ and Equation 3.7. Likewise, for fixed Θ^f, for any given x we can compute the expected fraction of locally functional relations R that are locally unambiguous for x.

Specifically, we repeat until convergence:

1. Set $\theta^f_R = \frac{1}{s_R} \sum_x P(R^*_x | D_{R(x,\cdot)}) \theta^u_x$ for all R.

2. Set $\theta^u_x = \frac{1}{s_x} \sum_R P(R^*_x | D_{R(x,\cdot)}) \theta^f_R$ for all x.

In both steps above, the sums are taken over only those x or R for which $D_{R(x,\cdot)}$ is non-empty. Also, the normalizer $s_R = \sum_x \theta^u_x$ and likewise $s_x = \sum_R \theta^f_R$.

By iteratively setting the parameters to the expectations in steps 1 and 2, we arrive at a good setting of the parameters.

The above algorithm is experimentally investigated in [51], showing that the technique effectively identifies functional relations, and can power effective contradiction detection.

3.6.3 Synonym Resolution

The last application of Urns we will discuss is that of resolving which strings refer to the same objects or relations. In text, the same object is often referred to by multiple distinct names—“U.S.” and “United States” each refer to the same country, for example. Likewise, relationships between objects are often expressed in multiple distinct ways (e.g., “x is the capital of y” and “x, capital of y”).

The Resolver system performs *Synonym Resolution* – taking as input a set of extracted tuples (as discussed above, e.g., `IsCapitalOf(D.C., United States)`) and returning a set of clusters, where each cluster contains coreferential object strings or relationship strings [60].

Here we provide a high-level description of how RESOLVER employs an URNS-like model, deferring to [60] for the details. Consider the task of determining whether two strings s_1 and s_2 refer to the same object, based on a set of tuples each including either s_1 or s_2 as an argument. RESOLVER specifies a urn-based generative process for the observed tuples; namely, the set of potential tuples for s_i are modeled as labels on balls in a urn, and the actual observed tuples involving s_i are modeled as draws from the urn. RESOLVER assumes that if s_1 and s_2 refer to the same object, then the urn contents for s_1 are maximally similar to those for s_2; otherwise, the two urns can differ to a greater or lesser degree. With this assumption, RESOLVER computes the probability that s_1 and s_2 co-refer based on how frequently they participate in similar tuples. This method is shown to be effective for resolving synonymous strings in practice.

3.7 Theoretical Results with URNS

In this section, we analyze the URNS model theoretically. To better understand the behavior of URNS, would like the be able to characterize how the probability that an example is a member of the target class increases with count MF values. Further, we would also like assurances that URNS can perform well, given enough unlabeled data. We would like to determine how accuracy increases with sample size, and most importantly prove that the parameters of the model can be learned from unlabeled data, and that in so doing URNS can overcome the limitations of the MFM.

Specifically, we investigate the following questions in the context of a single-urn model:

1. In the model, at what *rate* does the probability that an example is of the target class increase with the monotonic feature value?

2. What are sufficient conditions for accurate classification, given the parameters of the model? What sample size n is sufficient to achieve a given level of classification accuracy?

3. Can the parameters of the model be learned from unlabeled data?

4. Can the URNS model overcome the theoretical limitations of the MFM (discussed in Section 3.1) for unsupervised classification?

We begin by considering the first two questions in the uniform special case previously introduced in Section 3.2.1. The uniform case, while not fully realistic, does provide interpretable results; and we provide these for illustration. We then address all four questions in the more realistic Zipfian model used in our experiments.

In the below, for notational convenience we will utilize in place of the multi-set $num(C)$ a multi-set F_C containing, for each element of C, the relative *fraction* of balls labeled with that element. We define F_E similarly, such that $\sum_{f \in F_C \cup F_E} f = 1$. Then the following expression (adapted from Equation 12) specifies the probability that x is an element of C given the observed values of k and n:

$$P(x \in C|k, n) = \frac{\sum_{f \in F_C} f^k (1-f)^{n-k}}{\sum_{f \in F_C \cup F_E} f^k (1-f)^{n-k}}$$

3.7.1 Theoretical Results: Known Parameters

This section presents our theoretical results when the parameters of URNS are known. In the following, we examine URNS under two sets of assumptions, the Uniform Special Case (USC) and the Zipfian Case (ZC), defined below.

Theorems 14 and 16 address question (1) in each model, describing how class probability increases with the value k of the count MF. Specifically, we provide expressions for the increase in the odds ratio $odds(k, n) = P(x \in C|k, n)/(1 - P(x \in C|k, n))$ in terms of the count MF k. Theorems 15 and 18 address question (2). Let c_{known} indicate the classifier output by URNS when the parameters are known; we provide upper bounds on the expected error $E[error(c_{known})]$ in terms of the sample size n and the model parameters.

3.7.2 Analyzing the Uniform Special Case

The *Uniform Special Case* (USC) of the URNS model, first introduced in Section 3.2.1, is characterized by the following assumptions:

USC1 Each target label has the same probability p_C of being selected in a single draw, and each error label has a corresponding probability p_E.

USC2 The feature k is a count MF, so elements of C are repeated in the urn more frequently than elements of E (that is, $p_C > p_E$).

USC3 Frequency observations k are Poisson distributed (as in Equation 3.2).

Theoretical Results in the USC

The following theorem states how the odds ratio $odds(k, n)$ increases with k in the USC.

Theorem 14 *In the USC*

$$\frac{odds(k_1, n)}{odds(k_2, n)} = \left(\frac{p_C}{p_E}\right)^{k_1 - k_2} \tag{3.8}$$

Proof. Follows from the posterior probability in the USC (from Equation 3.2):

$$P(x \in C|k, n) = \frac{1}{1 + \frac{|E|}{|C|}(\frac{p_E}{p_C})^k e^{n(p_C - p_E)}} \tag{3.9}$$

Along with assumption USC2, Theorem 14 illustrates that in the USC the odds that an element is a member of the class increase exponentially with the value of the count MF. The increase is hastened when the target and error classes are less confusable (i.e. when $p_C >> p_E$).

How accurately we can perform classification, given the parameters of the model and the sample size? Let c_{known} indicate the Urns classifier when the parameters are known. The following theorem provides an upper bound on the error of c^* in the USC in terms of the sample size n, and the separability $p_C - p_E$ between the C and E sets.

Theorem 15 *In the USC, the expected error* $E[error(c_{\text{known}})] < \epsilon$ *when the sample size* n *satisfies:*

$$n \geq \frac{12 \ln 1/\epsilon}{(p_C - p_E)^2} \tag{3.10}$$

Proof Define a model m with a threshold $\tau = \frac{p_C + p_E}{2}$ such that $P_m(x \in C|k, n) \geq 0.5$ whenever $k \geq n\tau$, and $P_m(x \in C|k, n) < 0.5$ otherwise. Since τ is a sub-optimal threshold when the parameter values are known, $E[error(c_{\text{known}})]$ is bounded above by the expected error made by model m.We express the expected error of model m over the full set $C \cup E$ by summing the expected contribution of each individual term (equal to the probability that the term appears a number of times resulting in misclassification).

$$E[error(c_{\text{known}})] = \frac{\sum_{x\in E}\sum_{k\geq n\tau} P(k|x\in E, n) + \sum_{x\in C}\sum_{k<n\tau} P(k|x\in C, n)}{|C\cup E|} \tag{3.11}$$

Employing a Chernoff bound, we can bound the probability that a given term deviates from its expected frequency enough to be misclassified.

$$\begin{aligned}\sum_{k\geq n\tau} P(k|x\in E, n) &= \sum_{k\geq n(p_E+d/2)} P(k|x\in E, n) \\ &\leq e^{-nd^2/12}\end{aligned}$$

and

$$\begin{aligned}\sum_{k< n\tau} P(k|x\in C, n) &= \sum_{k< n(p_C-d/2)} P(k|x\in C, n) \\ &\leq e^{-nd^2/12}\end{aligned}$$

where $d = p_C - p_E$. Algebra gives the final result. □

Theorem 15 yields the following corollary, which states that under the assumptions of the USC, even a weakly indicative MF (one for which $p_C - p_E$ is just slightly greater than zero) can provide an arbitrarily accurate classifier, given sufficiently large n. This statement is akin to similar results in boosting algorithms in machine learning [53].

Corollary 15.1 *In the USC, for any $\epsilon > 0$, any count MF for which $p_C - p_E > 0$ can be used to achieve accuracy of $1 - \epsilon$ given sufficient sample size n.*

3.7.3 Analyzing the Zipfian Single-urn Case

The USC is a reasonable approximation for examples on the flat tail of the Zipf curve, but it is clearly an oversimplification for all examples. The following theorems are analogous to those presented for the USC above, but employ the more realistic Zipfian single-urn assumptions. In particular, we assume that the target and error sets are governed by known Zipf distributions, described below, with sizes $|C|$ and $|E|$ and shape parameters z_C and z_E. Further, we assume draws are generated from a mixture of these Zipf distributions, governed by a known mixing parameter p giving the probability that a single draw comes from C:

$$p = \sum_{f \in F_C} f \tag{3.12}$$

As in our experiments, we will find it convenient to work with a continuous representation of the commonly discrete Zipfian distribution. In the discrete Zipfian case, it is assumed that the ith most frequent element of C has frequency α_C / i^{z_C}, for α_C a normalization constant. In our continuous representation, the frequency of each element of C is itself a random variable drawn by choosing a uniform x from the range $[1, |C|+1]$ and then mapping x to the curve $f_C(x) = \alpha_C / x^{z_C}$ to obtain a frequency. The normalization constant α_C is:

$$\alpha_C = \frac{p}{\int_1^{|C|+1} \frac{1}{x^{z_C}} dx} \tag{3.13}$$

The normalization constant is chosen such that if we draw $|C|$ frequencies for the labels of the C set, the expected sum of the frequencies is p, as desired. The frequency of each element of E is defined analogously. We will refer to the functions f_C and f_E as *frequency curves.*

As in the USC, for a label in the ZC with underlying frequency f we assume the observed count k is Poisson distributed with expected value nf. Thus, the likelihood of observing an example of the C set k times in n draws is:

$$P_{ZC}(k | x \in C, n) = \frac{1}{|C|} \int_1^{|C|+1} \frac{(n\alpha_C x^{-z_C})^k}{e^{n\alpha_C x^{-z_C}} k!} dx \tag{3.14}$$

The solution of the above equation in terms of incomplete gamma functions is given below in Theorem 16.

We state the assumptions in the ZC as follows:

ZC1 The distributions of labels from C and E are each Zipfian as defined above, with mixing parameter p. Specifically, the likelihood of the data is governed by Equation 3.14.

ZC2 The observed frequency k is a count MF.

ZC3 The error label frequency curve has positive probability mass below the minimum target label frequency; that is $\alpha_E/i < \alpha_C/(|C|+1)$ for some known $i < |E|+1$.

ZC4 Analogously, the target label frequency curve has positive probability mass above the maximum error label frequency; that is $\alpha_C/i > \alpha_E$ for some known $i > 1$.

ZC5 Both the target and error set have non-zero probability mass in the urn; that is, $p, 1-p > M$ for some known lower bound M.

Assumptions ZC3 and ZC4 encode an assumption that given a sufficient number of distinct labels, with high probability the most frequent labels will be target labels and the least frequent will be error labels. These assumptions will allow us to establish PAC learnability from unlabeled data alone.

To lend justification to the above assumptions, we note that we would expect them to hold at least approximately in Unsupervised Information Extraction applications. The Zipfian nature of extractions and monotonicity (ZC1 and ZC2) are well known to hold approximately in practice. Further, assumption ZC3 is certainly empirically true when one considers that, as a simple example, for any target set element there exist multiple less-frequent misspellings in the error set. Assumption ZC4 tends to be at least approximately true in practice; the most frequently extracted labels tend to be instances of the target class. Assumption ZC5 is nearly trivially true in practice, we would always expect the target and error sets to have probability mass above some non-zero minimum value.

Theoretical Results in the ZC

We start by explicitly expressing how the odds that an element is a member of the target class increases with the value of the count MF:

Theorem 16 *In the ZC, the odds ratio*

$$\frac{odds(k+1,n)}{odds(k,n)} = \frac{(k-1/z_C)g(k,z_C,np,|C|+1,\alpha_C)+h(k-1/z_C,\frac{np}{(|C|+1)^{z_C}\alpha_C})}{(k-1/z_E)g(k,z_E,n(1-p),|E|+1,\alpha_E)+h(k-1/z_E,\frac{n(1-p)}{(|E|+1)^{z_E}\alpha_E})}$$

where

$$h(k',n') = n'^{k'}e^{n'}$$

and

$$P(k|x\in C,n) = \frac{np}{\alpha_C}^{1/z_C} g(k,z_C,np,|C|+1,\alpha_C) \tag{3.15}$$

with

$$g(k',z',n',s',\alpha') = \Gamma(k'-1/z',\frac{n'}{s'^{z'}\alpha'}) - \Gamma(k'-1/z',\frac{n'}{\alpha'})$$

assuming that neither z_C nor z_E are exactly equal to 1.

Proof. Given that $|C|,|E|,k \geq 1$, and $z_C, z_E \neq 1$ the above result is obtained by symbolic integration in Mathematica and algebra. □

Note that Theorem 16 does not utilize any assumptions other than the Zipfian mixture (ZC1). Equation 3.15 is the closed-form likelihood expression used to perform efficient inference in our experiments. Of course, the odds ratio given above is complex, and in order to provide qualitative insights should be simplified into a more interpretable bound; this is an item of future work.

We also wish to bound the classification error of Urns for the ZC. The following theorem provides a bound relative to the error of the *optimal classifier*, which utilizes both the Urns parameters *and* the precise frequencies of each label (rather than simply the observed counts). As such, the optimal classifier exhibits the best classification performance that can be achieved using the count MF alone.

Definition 17 *The* optimal classifier *is one which classifies each example optimally given knowledge of both the urn parameters, as well as the precise frequency in the urn of each example* $\mathbf{x} \in X$.

Define τ such that the classification threshold of the optimal classifier for a given n is equal to $n\tau$. From assumption ZC2, we know that a single such τ exists. Then the following theorem illustrates that as the sample size increases, the expected error falls off nearly linearly toward that of the optimal classifier.

Theorem 18 *In the ZC, given any $\delta > 0$, the expected error of urns is bounded as:*

$$E[error(c_{\text{known}})] \leq \beta + \frac{K_C(\delta) + K_E(\delta)}{|X|n^{1-\delta}}$$

where $K_C(\delta)$ and $K_E(\delta)$ are constants (with respect to n) defined below, and β is the expected error of the optimal classifier.

The constants K_C and K_E are defined as follows:

$$K_S(\delta) = \max\left(3/\delta, 1 + \int_1^{x_S^\tau - 1} \frac{1}{4(\alpha_S x^{-z_S} - x_S^\tau)^2} dx\right) \tag{3.16}$$

where x_S^τ is defined to be the unique value such that $f_S(x_S^\tau) = \tau$, and α_S is the normalization constant (see Equation 3.13).

Proof. Following the proof of Theorem 15, we aggregate the probabilities of the elements being misclassified.

We present the analysis for expected error on elements of the C set; the E set is analogous. We bound the probability that an element with true frequency of $\alpha_C x^{-z_C} > \tau$ appears fewer than $n\tau$ times in n draws using Chebyshev's inequality. Chebyshev's inequality bounds the probability that a random variable Y with expectation μ and variance σ^2 appears sufficiently far from its expectation:

$$P(|Y - \mu| > r\sigma) \leq \frac{1}{r^2}$$

For a Poisson random variable, σ is bounded above by $2\sqrt{n}$, so the above expression also bounds the probability that the deviation exceeds $2r\sqrt{n}$. Setting $2r\sqrt{n}$ equal to the smallest deviation resulting in misclassification ($n\alpha_C x^{-z_C} - nx_C^\tau$), and integrating over the frequency curve f_C, we have the following bound for the expected error on the C set:

$$E[error_C(c_{\text{known}})] \leq \beta_C + \int_1^{x_C^\tau} \min\left(1, \frac{1}{4n(\alpha_C x^{-z_C} - x_C^{\tau\,-z_C})^2}\right) dx \tag{3.17}$$

where β_C is the fraction of the expected risk of the optimal classifier due to elements of C (namely, the probability mass of elements of C with frequency less than τ).

Define:

$$\begin{aligned}\gamma_n &= \frac{1}{n} + \int_1^{x_C^\tau - 1/n} \frac{1}{4n(\alpha_C x^{-z_C} - x_C^{\tau\,-z_C})^2} dx \\ &\leq \int_1^{x_C^\tau} \min\left(1, \frac{1}{4n(\alpha_C x^{-z_C} - x_C^{\tau\,-z_C})^2}\right)\end{aligned}$$

We claim $\gamma_n \leq K_C(\delta)/n^{1-\delta}$, given which the theorem follows. The proof of the claim proceeds by induction. First, note that the $n = 1$ case, that $\gamma_1 \leq K_C(\delta)$, holds by construction of $K_C(\delta)$—the second term in the max function in Equation 3.16 is equal to γ_1. Then assuming $\gamma_n \leq K_C(\delta)/n^{1-\delta}$, consider the $n+1$ case:

$$\begin{aligned}\gamma_{n+1} &= \frac{n\gamma_n}{n+1} + \int_{x_C^\tau - 1/n}^{x_C^\tau - 1/(n+1)} \frac{1}{4n(\alpha_C x^{-z_C} - x_C^\tau)^2} dx \\ &\leq \frac{K_C(\delta) n^\delta}{n+1} + \frac{1}{n^2} \\ &= \frac{K_C(\delta)}{(n+1)^{1-\delta}} \left(\frac{n^\delta}{(n+1)^\delta} + \frac{(n+1)^{1-\delta}}{K_C(\delta) n^2}\right)\end{aligned}$$

It remains to show that:

$$\left(\frac{n^\delta}{(n+1)^\delta} + \frac{(n+1)^{1-\delta}}{K_C(\delta) n^2}\right) \leq 1 \tag{3.18}$$

With algebra, this is equivalent to the statement that $K_C(\delta) n^2((n+1)^\delta - n^\delta) \geq n+1$. From the generalized binomial theorem, $(n+1)^\delta$ is at least as large as $n^\delta + \delta n^{\delta-1} - \delta(1-\delta)n^{\delta-2}/2$. With algebra, we have:

$$K_C(\delta) n^2((n+1)^\delta - n^\delta) \geq \frac{K_C(\delta)\delta n^{1+\delta}}{2} \geq \frac{3n^{1+\delta}}{2} \geq n+1$$

as desired, using the fact that $K_C(\delta) \geq 3/\delta$. □

3.7.4 Theoretical Results with Unknown Parameters

In unsupervised classification, in general we are not given the Urns parameters in advance, and must learn these from unlabeled data. In this section, we provide theorems bounding the error in unsupervised classification even when the parameter values are unknown. The following theorem shows that with high probability the parameter values of Urns can be estimated accurately from unlabeled data alone, as the total number of examples $u = |C| + |E|$ increases, with n fixed.

Theorem 19 *In the ZC, for any $\delta, \epsilon > 0$, given $u = |C| + |E|$ examples for fixed n, we can obtain an estimate of the parameters of f_C and f_E such that with probability $1 - \delta$ each estimate lies within ϵ of the true parameter value.*

Proof. The frequency curves f_C and f_E can be converted into functions $g_C(\lambda)$ and $g_E(\lambda)$ giving the probability density of a particular frequency λ for labels in the C (resp. E) set. These functions are themselves power law distributions. For example, in the error set case:

$$g_E(\lambda) = \begin{cases} \frac{L_E}{\lambda^{(1+z_E)/z_E}} & \text{for } a_E \leq x \leq b_E, \\ 0 & \text{for } x < a_E \text{ or } x > b_E, \end{cases} \tag{3.19}$$

for a suitable constant L_E where z_E indicates the exponent from the original frequency curve. The distribution of error labels in the model is completely characterized by four parameters: L_E and z_E, the minimal frequency a_E, and the maximal frequency b_E.

The probability that a particular label appears k times can then be written as follows:

$$P(k|x \in C, n) = \int_0^n (g_C(\lambda) + g_E(\lambda)) \frac{e^{-\lambda}\lambda^k}{k!} d\lambda \tag{3.20}$$

Let $g(x) = g_C(x) + g_E(x)$. When written in the form of Equation 3.20, the distribution over k becomes an instance of a *compound Poisson process*, for which the existence of effective estimators of $g(x)$ is well-known. In particular, Theorem 1 from [36] states that for any $x < n$ we can obtain a sequence of estimates $\hat{g}_u(x)$ of $g(x)$ such that $E[\hat{g}_u(x) - g(x)]^2 = o(1)$ as $u \to \infty$. Thus, for any given $\delta, \epsilon' > 0$, we have with probability $1 - \delta$ that $|\hat{g}_u(x) - g_(x)| < \epsilon'$ for u sufficiently large. Our task is to convert this estimator of $g(x)$ into estimators of each of the Urns parameters. In the re-written model (Equation 3.19) we will employ, there are eight total parameters characterizing the mixture components g_C and g_E. We present the construction for the three parameters of g_E, the g_C case is analogous.

Consider two estimates $\hat{g}_u(x_0)$ and $\hat{g}_u(rx_0)$ where $x_0, rx_0 < \alpha_C/(|C| + 1)$. That is, x_0 and rx_0 are sufficiently small that $g_C(x_0)$ and $g_C(rx_0)$ are zero by assumption ZC3. By algebra, in this region $(1 + z_E)/z_E = (\ln g(x_0) - \ln g(rx_0))/(\ln r)$, so z_E is a continuous and bounded function of $g(x_0)$ and $g(rx_0)$ on the domain of interest. This implies we can estimate z_E within ϵ with probability $1-\delta$ given our estimator $\hat{g}_u$, for u suitably large. Likewise, L_E is a continuous and bounded function of $g(x)$ and z_E, so we can estimate L_E effectively.

It remains to obtain an estimator for the limits of support a_E and b_E. We begin with the minimal limit a_E. We construct from 0 to n a uniform lattice of estimates $\{\hat{g}_u(x_i)\}$ each ϵ apart. By assumption ZC5, $g_E(x) > M'$ for $x \in [a_E, a_E+\epsilon)$ for a known constant M' given that ϵ is sufficiently small. By taking u suitably large, we can ensure with probability $1-\delta$ that $\forall x_i < a_E$, $|\hat{g}_u(x_i) - 0| < M'/2$ and that the $x_j \geq a_E$ that falls in the interval $[a_E, a_E+\epsilon)$ has estimate $\hat{g}_u(x_j) > M'/2$. Thus, the minimal x_i such that $\hat{g}_u(x) > M'/2$ is with probability $1-\delta$ an estimate within ϵ of a_E. Estimating the maximal limit of support b_E is similar. The same procedure can be employed, except that because $g_C(b_E)$ is non-zero, we instead identify successive estimates $\hat{g}_u(x_k)$ and $\hat{g}_u(x_{k+1})$ that differ by a sufficiently large margin, where x_k is greater than our estimate for a_E. For u sufficiently large and ϵ sufficiently small, with probability $1-\delta$ the value x_k is within ϵ of b_E.

$\square$

Theoretical Results using All Features

In this section, we prove that Urns can overcome the limitations of the MFM discussed in Section 3.1.

We prove that even when the conditional independence assumption does not hold, a sufficiently informative count MF that follows the Urns model can be used to PAC learn from only unlabeled data, provided a "separability" criterion holds on the concept class $\mathcal{C}_{\neg M}$ given the MF. This criterion requires that no two distinct concepts in $\mathcal{C}$ for which x_i is an MF agree on too large a fraction of the instance space:

Definition 20 *A concept class $\mathcal{C}'$ is* **ϵ-separable for MF** *x_i if for any distinct concepts $c, c' \in \mathcal{C}'$ for which x_i is an MF, the fraction of examples $\mathbf{x} \in \mathcal{X}$ such that $c(\mathbf{x}) = c'(\mathbf{x})$ is less that $1-\epsilon$.*

We also require a count MF that is sufficiently informative. We state this criteria in terms of the minimal expected classification error that can be achieved with the count MF, in the limit of u and n large. This is equivalent to the area of the "confusion region" in Figure 3.1, which we define formally as:

Definition 21 *The* area of the confusion region *of a count MF is:*

$$\min_{\tau} \left[\int_0^{\tau} g_C(\lambda) d\lambda + \int_{\tau}^{\infty} g_E(\lambda) d\lambda \right] \tag{3.21}$$

Given this definition, we can state the following result, which shows that Urns is able to overcome the limitations of the MFM.

Proposition 22 *If $\mathcal{C}$ is ϵ-separable for MF x_i, given that x_i is an MF that follows the ZC with confusion region of area less than $1-\epsilon/2$, $\mathcal{C}$ is PAC-learnable from unlabeled data alone.*

Proof. By Theorem 19, we can obtain the parameters of Urns within ϵ' accuracy. Because the optimal classification threshold τ is a continuous and bounded function of the Urns parameters (see Equation 3.21), Urns can achieve accuracy arbitrarily close to the confusion region size of less than $1 - \epsilon/2$, provided n and u are sufficiently large. Thus, the accuracy of the Urns-based classifier is such that it assigns classifications different from those of the target classifier on fewer than $1 - \epsilon/2$ of the examples. By the separability criterion, the target concept is the only hypothesis compatible with the MF that differs from the output of Urns on so few examples. Thus, an algorithm that returns the concept $c \in \mathcal{C}$ most similar to the output of Urns will always return target concept. $\square$

3.8 Related Work

In contrast to the bulk of previous IE work, our focus is on unsupervised IE (UIE) where Urns substantially outperforms previous methods (Figure 3.2).

In addition to the noisy-or models we compare against in our experiments, the IE literature contains a variety of heuristics using repetition as an indication of the veracity of extracted information. For example, Riloff and Jones [50] rank extractions by the number of distinct patterns generating them, plus a factor for the reliability of the patterns. Our work is intended to formalize these heuristic techniques, and unlike the noisy-or models, we explicitly model the distribution of the target and error sets (our $num(C)$ and $num(E)$), which is shown to be important for good performance in Section 3.4.1. The accuracy of the probability estimates produced by the heuristic and noisy-or methods is rarely evaluated explicitly in the IE literature, although most systems make implicit use of such estimates. For example, bootstrap-learning systems start with a set of seed instances of a given relation, which are used to identify extraction patterns for the relation; these patterns are in turn used to extract further instances (e.g. [50, 35, 3]). As this process iterates, random extraction errors result in overly general extraction patterns, leading the system to extract further erroneous instances. The more accurate estimates of extraction probabilities produced by URNS would help prevent this "concept drift."

Skounakis and Craven [55] develop a probabilistic model for combining evidence from multiple extractions in a supervised setting. Their problem formulation differs from ours, as they classify each occurrence of an extraction, and then use a binomial model along with the false positive and true positive rates of the classifier to obtain the probability that at least one occurrence is a true positive. Similar to the above approaches, they do not explicitly account for sample size n, nor do they model the distribution of target and error extractions.

Culotta and McCallum [16] provide a model for assessing the confidence of extracted information using conditional random fields (CRFs). Their work focuses on assigning accurate confidence values to individual occurrences of an extracted field based on textual features. This is complementary to our focus on *combining* confidence estimates from multiple occurrences of the same extraction. In fact, each possible feature vector processed by the CRF in [16] can be thought of as a virtual urn m in our URNS. The confidence output of Culotta and McCallum's model could then be used to provide the precision p_m for the urn.

Our work is similar in spirit to BLOG, a language for specifying probability distributions over sets with unknown objects [40]. As in our work, BLOG models can express observations as draws from an unknown set of balls in an urn. Whereas BLOG is intended to be a general modeling framework for probabilistic first-order logic with varying sets of objects, our work is directed at modeling redundancy in IE. We also provide supervised and unsupervised learning methods for our model that are effective for data sets containing many thousands of examples, along with experiments demonstrating their efficacy in practice.

Joachims provides theoretical results in supervised textual classification that use the Zipfian structure of text to arrive at error bounds for Support Vector Machine classifiers on textual data [34]. The strong performance of SVMs in our supervised experiments corroborate Joachims's claim that these classifiers are effective on textual data. However, in contrast to Joachims's work, our experiments and theoretical results are focused on the unsupervised case. We show that when the Zipfian structure holds, unsupervised learning is possible under certain assumptions.

One of the problems our EM-based algorithm for learning Urns parameters must solve is estimating the parameter $|C|$, the size of the target set. This problem has commonalities with the classic "capture-recapture" problem from ecology, in which the goal is to estimate the size of an animal population by capturing and marking a sample of the population, then re-sampling at a later time. There are a number of significant differences between the capture-recapture problem and estimating Urns parameters, however. First, Urns attempts to learn the parameter $|C|$ from observations which are co-mingled with samples from a confounding error distribution. Second, Urns must also characterize how the frequencies of the target set vary (in terms of the Zipfian shape parameter z_C). In order to solve these more difficult parameter estimation problems, Urns exploits problem structures often found in textual domains, such as the fact that extractions and errors tend to be Zipf distributed.

3.9 Conclusions and Future Work

This chapter introduced a combinatorial Urns model to the problem of assessing the probability that an extraction is correct. The chapter described supervised and unsupervised methods for estimating the parameters of the model from data, and reported on experiments showing that Urns massively outperforms previous methods in the unsupervised case, and is slightly better than baseline methods in the supervised case. We also provided theoretical results proving that the Urns model can be effective in cases where the MFM alone is not, and established that the parameters of Urns can be learned from unlabeled data alone, given certain conditions. Our experimental and theoretical results are summarized below.

1. **Experimental Results in IE.** Listed below is Urns performance advantage against each baseline, for supervised, semi-supervised, and unsupervised IE. For (a) and (c), error reduction is measured in terms of deviation from ideal log likelihood. For (b), error reduction is measured in terms of classification error.

   (a) **Supervised IE** (Table 3.3)

   - 19% error reduction over noisy-or
   - 10% error reduction over logistic regression
   - Comparable performance to SVM

   (b) **Semi-supervised IE** (Figure 3.4)

   - 6% error reduction over LP

   (c) **Unsupervised IE** (Figure 3.2)

   - 1500% error reduction over noisy-or
   - 2000% error reduction over Pointwise Mutual Information (PMI)

2. **Theoretical Results in Zipfian Single-urn Case (ZC)** Below is an informal summary of the theoretical results for the ZC model.

   (a) **Theorem 16**

   - States how the odds of correctness increase with the MF value.

   (b) **Theorem 18**

   - The classification error of the model is less than a constant factor times $\frac{1}{1/n^{(1-\delta)}}$ for any $\delta > 0$ (n denotes the sample size).

   (c) **Theorem 19**

   - Parameters of the model are learnable from unlabeled data alone, given sufficient data.

   (d) **Theorem 22**

- Probably Approximately Correct (PAC) classification is possible in the model, given sufficient data.

Several items of future work are possible. One important direction is developing a probabilistic model for multiple mechanisms that is more flexible than multiple urns. The correlation model used for multiple urns is limited and can only handle a pre-defined set of distinct mechanisms. A method for UIE that leverages all contexts rather than a select set of extraction patterns is investigated in the next chapter; but it is not a probabilistic model and only ranks extractions, rather than producing probabilities or classifications. Lastly, the approximate "M step" we employ for unsupervised learning of Urns parameters could be improved upon, utilizing exact methods or more rigorous approximations.

Chapter 4
LANGUAGE MODELS FOR ASSESSING SPARSE EXTRACTIONS

Even in a massive corpus such as the Web, a substantial fraction of extractions appear infrequently. While the Urns model introduced in the previous chapter can assign accurate probabilities of correctness to these *sparse* extractions, it cannot distinguish which of these extractions are correct.

This chapter shows how to assess the correctness of sparse extractions by utilizing unsupervised language models. The Realm system, which combines HMM-based and n-gram-based language models, ranks candidate extractions by the likelihood that they are correct. Our experiments show that Realm reduces extraction error by 39%, on average, when compared with previous work.

Because Realm pre-computes language models based on its corpus and does not require *any* hand-tagged seeds, it is far more scalable than approaches that learn models for each individual relation from hand-tagged data. Thus, Realm is ideally suited for *open* information extraction, a special case of UIE in which the relations of interest are not specified in advance and their number is potentially vast.

4.1 Introduction

Zipf's Law governs the distribution of extractions. Thus, even the Web has limited redundancy for less prominent instances of relations. Indeed, 50% of the extractions in the data sets employed in Chapters 2 and 3 appeared only once. As a result, the Urns model (and related methods) have no way of assessing which extraction is more likely to be correct for fully half of the extractions. This problem is particularly acute when moving beyond unary relations. We refer to this challenge as the task of *assessing sparse extractions*.

This chapter introduces the idea that *language modeling* techniques such as n-gram statistics [37] and HMMs [46] can be used to effectively assess sparse extractions. The chapter introduces the REALM system, and highlights its unique properties. Notably, REALM does not require *any* hand-tagged seeds, which enables it to scale to *Open IE*—a special case of UIE where the relations of interest are not specified in advance, and their number is potentially vast [5].

REALM is based on two key hypotheses. The *KnowItAll hypothesis*, introduced in Chapter 1, states that extractions that occur more frequently in distinct sentences in the corpus are more likely to be correct. For example, the hypothesis suggests that the argument pair (`Giuliani`, `New York`) is relatively likely to be appropriate for the `Mayor` relation, simply because this pair is extracted for the `Mayor` relation relatively frequently. Second, we employ an instance of the *distributional hypothesis* [32], which can be phrased as follows: different instances of the same semantic relation tend to appear in similar textual contexts. We assess sparse extractions by comparing the contexts in which they appear to those of more common extractions. Sparse extractions whose contexts are more similar to those of common extractions are judged more likely to be correct based on the conjunction of the KnowItAll and the distributional hypotheses.

The contributions of the chapter are as follows:

- The chapter introduces the insight that the subfield of language modeling provides unsupervised methods that can be leveraged to assess sparse extractions. These methods are more scalable than previous assessment techniques, and require no hand tagging whatsoever.

- The chapter introduces an HMM-based technique for checking whether two arguments are of the proper type for a relation.

- The chapter introduces a *relational* n-gram model for the purpose of determining whether a sentence that mentions multiple arguments actually expresses a particular relationship between them.

- The chapter introduces a novel language-modeling system called REALM that combines both HMM-based models and relational n-gram models, and shows that REALM reduces error by an average of 39% over previous methods, when applied to sparse extraction data.

The remainder of the chapter is organized as follows. Section 4.2 introduces the IE assessment task, and describes the Realm system in detail. Section 4.3 reports on our experimental results followed by a discussion of related work in Section 4.4. Finally, we conclude with a discussion of scalability and with directions for future work.

4.2 IE Assessment

This section formalizes the *IE assessment* task, which has important differences from the IE classification tasks considered in the previous chapters, and then describes the Realm system for solving it. An IE assessor takes as input a list of candidate extractions meant to denote instances of a relation, and outputs a *ranking* of the extractions with the goal that correct extractions rank higher than incorrect ones. A *correct* extraction is defined to be a true instance of the relation mentioned in the input text.

More formally, the list of candidate extractions for a relation R is denoted as $E_R = \{(a_1, b_1), \ldots, (a_m, b_m)\}$. An extraction $(a_i,\, b_i)$ is an ordered pair of strings. The extraction is *correct* if and only if the relation R holds between the arguments named by a_i and b_i. For example, for $R = \texttt{Headquartered}$, a pair (a_i, b_i) is correct *iff* there exists an organization a_i that is in fact headquartered in the location b_i.[1]

E_R is generated by applying an extraction mechanism, typically a set of extraction "patterns", to each sentence in a corpus, and recording the results. Thus, many elements of E_R are identical extractions derived from different sentences in the corpus.

This task definition is notable for the minimal inputs required—IE assessment does not require knowing the relation name nor does it require hand-tagged seed examples of the relation. Thus, an IE Assessor is applicable to Open IE.

4.2.1 System Overview

In this section, we describe the Realm system, which utilizes language modeling techniques to perform IE Assessment.

[1]For clarity, our discussion focuses on relations between pairs of arguments. However, the methods we propose can be extended to relations of any arity.

REALM takes as input a set of extractions E_R, and outputs a ranking of those extractions. The algorithm REALM follows is outlined in Figure 4.1. Similar to the MFA algorithm from Chapter 2, REALM begins by automatically selecting from E_R a set of *bootstrapped seeds* S_R intended to serve as correct examples of the relation R. REALM utilizes the KnowItAll hypothesis, setting S_R equal to the h elements in E_R extracted most frequently from the underlying corpus. This results in a noisy set of seeds, but the methods that use these seeds are noise tolerant.

REALM then proceeds to rank the remaining (non-seed) extractions by utilizing two language-modeling components. An *n-gram language model* is a probability distribution $P(w_1, ..., w_n)$ over consecutive word sequences of length n in a corpus. Formally, if we assume a seed (s_1, s_2) is a correct extraction of a relation R, the distributional hypothesis states that the *context distribution* around the seed extraction, $P(w_1, ..., w_n | w_i = s_1, w_j = s_2)$ for $1 \leq i, j \leq n$ tends to be "more similar" to $P(w_1, ..., w_n | w_i = e_1, w_j = e_2)$ when the extraction (e_1, e_2) is correct. Naively comparing context distributions is problematic, however, because the arguments to a relation often appear separated by several intervening words. In our experiments, we found that when relation arguments appear together in a sentence, 75% of the time the arguments are separated by at least three words. This implies that n must be large, and for sparse argument pairs it is not possible to estimate such a large language model accurately, because the number of modeling parameters is proportional to the vocabulary size raised to the nth power. To mitigate sparsity, REALM utilizes smaller language models in its two components as a means of "backing-off' from estimating context distributions explicitly, as described below.

First, REALM utilizes an HMM to estimate whether each extraction has arguments of the proper type for the relation. Each relation R has a set of types for its arguments. For example, the relation `AuthorOf(a, b)` requires that its first argument be an author, and that its second be some kind of written work. Knowing whether extracted arguments are of the proper type for a relation can be quite informative for assessing extractions. The challenge is, however, that this type information is *not* given to the system since the relations (and the types of the arguments) are not known in advance. REALM solves this problem by comparing the distributions of the seed arguments and extraction arguments. Type checking mitigates data sparsity by leveraging *every* occurrence of the individual extraction arguments in the corpus, rather than only those cases in which argument pairs occur near each other.

Although argument type checking is invaluable for extraction assessment, it is not sufficient for extracting relationships between arguments. For example, an IE system using only type information might determine that `Intel` is a corporation and that `Seattle` is a city, and therefore erroneously conclude that `Headquartered(Intel, Seattle)` is correct. Thus, REALM's second step is to employ an n-gram-based language model to assess whether the extracted arguments share the appropriate relation. Again, this information is not given to the system, so REALM compares the context distributions of the extractions to those of the seeds. As described in Section 4.2.3, REALM employs a relational n-gram language model in order to accurately compare context distributions when extractions are sparse.

REALM executes the type checking and relation assessment components separately; each component takes the seed and non-seed extractions as arguments and returns a ranking of the non-seeds. REALM then combines the two components' assessments into a single ranking. Although several such combinations are possible, REALM simply ranks the extractions in ascending order of the product of the ranks assigned by the two components. The following subsections describe REALM's two components in detail.

We identify the proper nouns in our corpus using the LEX method [18]. In addition to locating the proper nouns in the corpus, LEX also concatenates each multi-token proper noun (*e.g.*,`Los Angeles`) together into a single token. Both of REALM's components construct language models from this tokenized corpus.

4.2.2 Type Checking with HMM-T

In this section, we describe our type-checking component, which takes the form of a Hidden Markov Model and is referred to as HMM-T. HMM-T ranks the set U_R of non-seed extractions, with a goal of ranking those extractions with arguments of proper type for R above extractions containing type errors. Formally, let U_{Ri} denote the set of the ith arguments of

```
MFA($Extractions\ E_R = \{e_1, ..., e_m\}$)
    $S_R$ = the $h$ most frequent extractions in $E_R$
    $U_R = E_R$ - $S_R$
    $TypeRankings(U_R) \leftarrow$ HMM-T($S_R$, $U_R$)
    $RelationRankings(U_R) \leftarrow$ REL-GRAMS($S_R$, $U_R$)
    return a ranking of $E_R$ with the elements of $S_R$ at the
        top (ranked by frequency) followed by the elements of
        $U_R = \{u_1, ..., u_{m-h}\}$ ranked in ascending order of
        $TypeRanking(u_i) * RelationRanking(u_i)$.
```

Figure 4.1: Pseudocode for MFA at run-time. The language models used by the HMM-T and REL-GRAMS components are constructed in a pre-processing step.

the extractions in U_R. Let S_{Ri} be defined similarly for the seed set S_R.

Our type checking technique exploits the distributional hypothesis—in this case, the intuition that extraction arguments in U_{Ri} of the proper type will likely appear in contexts similar to those in which the seed arguments S_{Ri} appear. In order to identify terms that are distributionally similar, we train a probabilistic generative Hidden Markov Model (HMM), which treats each token in the corpus as generated by a single hidden state variable. Here, the hidden states take integral values from $\{1, \ldots, T\}$, and each hidden state variable is itself generated by some number k of previous hidden states.[2] Formally, the joint distribution of the corpus, represented as a vector of tokens $\mathbf{w}$, given a corresponding vector of states $\mathbf{t}$ is:

$$P(\mathbf{w}|\mathbf{t}) = \prod_i P(w_i|t_i)P(t_i|t_{i-1}, \ldots, t_{i-k}) \tag{4.1}$$

The distributions on the right side of Equation 4.1 can be learned from a corpus in an unsupervised manner, such that words which are distributed similarly in the corpus tend to be generated by similar hidden states [46]. The generative model is depicted as a Bayesian network in Figure 4.2. The figure also illustrates the one way in which our implementation

[2]Our implementation makes the simplifying assumption that each sentence in the corpus is generated independently.

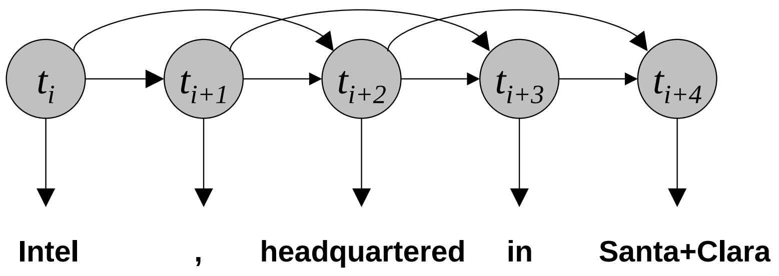

Figure 4.2: Graphical model employed by HMM-T. Shown is the case in which $k = 2$. Corpus pre-processing results in the proper noun `Santa Clara` being concatenated into a single token.

is distinct from a standard HMM, namely that proper nouns are detected *a priori* and modeled as single tokens (*e.g.*, `Santa Clara` is generated by a single hidden state). This allows the type checker to compare the state distributions of different proper nouns directly, even when the proper nouns contain differing numbers of words.

To generate a ranking of U_R using the learned HMM parameters, we rank the arguments e_i according to how similar their *state distributions* $P(t|e_i)$ are to those of the seed arguments.[3] Specifically, we define a function:

$$f(e) = \sum_{e_i \in e} KL\left(\frac{\sum_{w' \in S_{Ri}} P(t|w')}{|S_{Ri}|}, P(t|e_i)\right) \tag{4.2}$$

where KL represents KL divergence, and the outer sum is taken over the arguments e_i of the extraction e. We rank the elements of U_R in ascending order of $f(e)$.

HMM-T has two advantages over a more traditional type checking approach of simply counting the number of times in the corpus that each extraction appears in a context in which a seed also appears (*cf.* [47]). The first advantage of HMM-T is efficiency, as the traditional approach involves a computationally expensive step of retrieving the potentially large set of contexts in which the extractions and seeds appear. In our experiments, using HMM-T instead of a context-based approach results in a 10-50x reduction in the amount of data that is retrieved to perform type checking. Secondly, on sparse data HMM-T has the potential to improve type checking accuracy. For example, consider comparing `Pickerington`, a

[3]The distribution $P(t|e_i)$ for any e_i can be obtained from the HMM parameters using Bayes Rule.

sparse candidate argument of the type `City`, to the seed argument `Chicago`, for which the following two phrases appear in the corpus:

(i) "Pickerington, Ohio"

(ii) "Chicago, Illinois"

In these phrases, the textual contexts surrounding `Chicago` and `Pickerington` are not identical, so to the traditional approach these contexts offer no evidence that `Pickerington` and `Chicago` are of the same type. For a sparse token like `Pickerington`, this is problematic because the token may *never* occur in a context that precisely matches that of a seed. In contrast, in the HMM, the non-sparse tokens `Ohio` and `Illinois` are likely to have similar state distributions, as they are both the names of U.S. States. Thus, in the state space employed by the HMM, the contexts in phrases (i) and (ii) are in fact quite similar, allowing HMM-T to detect that `Pickerington` and `Chicago` are likely of the same type. Our experiments quantify the performance improvements that HMM-T offers over the traditional approach for type checking sparse data.

Each iteration required to learn HMM-T's parameters takes time proportional to T^{k+1} times the corpus size. Thus, for tractability, HMM-T uses a relatively small state space of $T = 20$ states and a limited k value of 3. While these settings are sufficient for type checking (*e.g.*, determining that `Santa Clara` is a city) they are too coarse-grained to assess relations between arguments (*e.g.*, determining that `Santa Clara` is the particular city in which `Intel` is headquartered). We now turn to the REL-GRAMS component, which performs the latter task.

4.2.3 Relation Assessment with REL-GRAMS

REALM's relation assessment component, called REL-GRAMS, tests whether the extracted arguments have a desired relationship, but given REALM's minimal input it has no *a priori* information about the relationship. REL-GRAMS relies instead on the distributional hypothesis to test each extraction.

As argued in Section 4.2.1, it is intractable to build an accurate language model for context distributions surrounding sparse argument pairs. To overcome this problem, we introduce *relational* n-gram models. Rather than simply modeling the context distribution around a given argument, a relational n-gram model specifies separate context distributions for an arguments *conditioned on* each of the other arguments with which it appears. The relational n-gram model allows us to estimate context distributions for pairs of arguments, even when the arguments do not appear together within a fixed window of n words. Further, by considering only consecutive argument pairs, the number of distinct argument pairs in the model grows at most linearly with the number of sentences in the corpus. Thus, the relational n-gram model can scale.

Formally, for a pair of arguments (e_1, e_2), a relational n-gram model estimates the distributions $P(w_1, ..., w_n | w_i = e_1, e_1 \leftrightarrow e_2)$ for each $1 \leq i \leq n$, where the notation $e_1 \leftrightarrow e_2$ indicates the event that e_2 is the next argument to either the right or the left of e_1 in the corpus.

REL-GRAMS begins by building a relational n-gram model of the arguments in the corpus. For notational convenience, we represent the model's distributions in terms of "context vectors" for each pair of arguments. Formally, for a given sentence containing arguments e_1 and e_2 consecutively, we define a *context* of the ordered pair (e_1, e_2) to be any window of n tokens around e_1.[4]

Let $C = \{c_1, c_2, ..., c_{|C|}\}$ be the set of all contexts of all argument pairs found in the corpus.[5] For a pair of arguments (e_j, e_k), we model their relationship using a $|C|$ dimensional context vector $v_{(e_j, e_k)}$, whose i-th dimension corresponds to the number of times context c_i occurred with the pair (e_j, e_k) in the corpus. These context vectors are similar to document vectors from Information Retrieval (IR), and we leverage IR techniques to compare them, as described below.

To assess each extraction, we determine how similar its context vector is to a canonical seed vector (created by summing the context vectors of the seeds). While there are many

[4]In practice, when computing contexts we replace each entity with an appropriate special token. For example, from the sentence "This information came from Intel in Santa+Clara", using a window of length $n = 3$, we find three contexts for the entity pair (`Intel`, `Santa+Clara`): 'came from `OBJ`', 'from `OBJ` in', and '`OBJ` in `RELATED`'.

[5]Pre-computing the set C requires identifying in advance the potential relation arguments in the corpus. We consider the proper nouns identified by the LEX method (see Section 4.2.1) to be the potential arguments.

potential methods for determining similarity, in this work we rank extractions by decreasing values of the BM25 distance metric. BM25 is a TF-IDF variant introduced in TREC-3[52], which outperformed both the standard cosine distance and a smoothed KL divergence on our data. The BM25 distance between the extraction context vector e and seed context vector s is defined as:

$$BM25(e,s) = \sum_{1 \leq j \leq |C|} \frac{(k_1+1) * e_j}{k_1((1-b) + b * \frac{cl}{acl}) + e_j} w_j \tag{4.3}$$

$$w_j = \log \frac{(r_j + 0.5) * (N - n_j - R + r_j + 0.5)}{(R - r_j + 0.5) * (n_j - r_j + 0.5)} \tag{4.4}$$

where R is the total number of seeds, r_j is the number of seeds observed at least once with context j, N is the total number of entity pairs, n_j is the total number of entity pairs seen at least once in context j, cl is the total number of contexts e appears with, acl is the average number of contexts an extraction appears with, and k_1 and b are tuning parameters.

For all our experiments we set $k_1 = 3$ and $b = 1$ and used a $t = 4$ token window for contexts. We examined several variations of these parameters and found that the results were not sensitive to these parameters. The relational n-gram model captures e_1's relationship to e_2 separately from e_2's relationship to e_1. In our implementation, REL-GRAMS utilizes a symmetric measure, equal to the sum of the BM25 scores computed for each direction.

4.3 Experimental Results

This section describes our experiments on IE assessment for sparse data. We start by describing our experimental methodology, and then present our results. The first experiment tests the hypothesis that HMM-T outperforms an n-gram-based method on the task of type checking. The second experiment tests the hypothesis that REALM outperforms multiple approaches from previous work, and also outperforms each of its HMM-T and REL-GRAMS components taken in isolation.

4.3.1 Experimental Methodology

The corpus used for our experiments consisted of a sample of sentences taken from Web pages. From an initial crawl of nine million Web pages, we selected sentences containing relations between proper nouns. The resulting text corpus consisted of about three million sentences, and was tokenized as described in Section 4.2. For tractability, before and after performing tokenization, we replaced each token occurring fewer than five times in the corpus with one of two "unknown word" markers (one for capitalized words, and one for uncapitalized words). This preprocessing resulted in a corpus containing about sixty-five million total tokens, and 214,787 unique tokens.

We evaluated performance on four relations: `Conquered`, `Founded`, `Headquartered`, and `Merged`. These four relations were chosen because they typically take proper nouns as arguments, and included a large number of sparse extractions. For each relation R, the candidate extraction list E_R was obtained using TextRunner [5]. TextRunner computes an index of all extracted relationships it recognizes, in the form of (object, predicate, object) triples. For each of our target relations, we executed a single query to the TextRunner index for extractions whose predicate contained a phrase indicative of the relation (*e.g.*, "founded by", "headquartered in"), and the results formed our extraction list. For each relation, the 10 most frequent extractions served as bootstrapped seeds. All of the non-seed extractions were sparse (no argument pairs were extracted more than twice for a given relation). These test sets contained a total of 361 extractions.

4.3.2 Type Checking Experiments

As discussed in Section 4.2.2, on sparse data Hmm-T has the potential to outperform type checking methods that rely on textual similarities of context vectors. To evaluate this claim, we tested the Hmm-T system against an n-grams type checking method on the task of type-checking the arguments to a relation. The n-grams method compares the context vectors of extractions in the same way as the Rel-grams method described in Section 4.2.3, but is not relational (n-grams considers the distribution of each extraction argument independently, similar to Hmm-T). We tagged an extraction as type correct *iff* both arguments were valid for the relation, ignoring whether the relation held between the arguments.

The results of our type checking experiments are shown in Table 4.1. For all types, Hmm-

Table 4.1: Type Checking Performance. Listed is area under the precision/recall curve. HMM-T outperforms N-GRAMS for all relations, and reduces the error in terms of missing area under the curve by 46% on average.

Type	HMM-T	N-GRAMS
`Conquered`	**0.917**	0.767
`Founded`	**0.827**	0.636
`Headquartered`	**0.734**	0.589
`Merged`	**0.920**	0.854
Average	**0.849**	0.712

T outperforms N-GRAMS, and HMM-T reduces error (measured in missing area under the precision/recall curve) by 46%. The performance difference on each relation is statistically significant ($p < 0.01$, two-sampled t-test), using the methodology for measuring the standard deviation of area under the precision/recall curve given in [48]. N-GRAMS, like REL-GRAMS, employs the BM-25 metric to measure distributional similarity between extractions and seeds. Replacing BM-25 with cosine distance cuts HMM-T's advantage over N-GRAMS, but HMM-T's error rate is still 23% lower on average.

Additional Experiments with HMM-T

The type checking task is similar to the UIE task for unary extractions considered in Chapters 2 and 3. Can HMM-T improve performance on unary extractions as well?

We obtained candidate extractions of each unary relation by searching a corpus for patterns likely to indicate the type (e.g., "countries such as x"). For a given type, the most frequent extractions occurring more than once (to a maximum of 10) served as seed examples. To focus our experiments on the assessment of sparse data, our test sets were composed of all extractions co-occurring with the patterns only once. In our corpus, the singleton extractions numbered 99 for `Company`, 50 for `Country`, 77 for `Film`, and 39 for `Language`.

Table 4.2 shows the performance of both HMM-T and N-GRAMS on three classes. Here,

Table 4.2: Performance of N-GRAMS and HMM-T on four classes from the UIE task. HMM-T reduces error by 28% on average.

Class	HMM-T	N-GRAMS
`Company`	**0.966**	0.956
`Country`	**0.886**	0.800
`Film`	**0.803**	0.741
`Language`	**0.936**	0.931
Average	**0.898**	0.857

we compare against the better-performing cosine metric for N-GRAMS, rather than BM25. HMM-T reduces error (in terms of missing area under the precision/recall curve) by 28% on average over N-GRAMS.

Another question is to what extent the noisy seeds that HMM-T uses impact its performance. We investigated this question is the context of the unary extraction experiments above. First, only the `Film` class in these experiments had seed extractions which were errors; thus, we see that HMM-T can improve performance over N-GRAMS even when the seeds are known to be correct. When the five erroneous seeds for the `Film` class are removed, we find that this reduces error of the HMM-T method by 8% (increasing area under the precision/recall curve to 0.819, from 0.803) versus a larger 14% error reduction for the N-GRAMS method (to 0.776, from 0.741). This suggests that HMM-T is more robust than N-GRAMS when seed extractions are noisy.

4.3.3 Experiments with REALM

The REALM system combines the type checking and relation assessment components to assess extractions. Here, we test the ability of REALM to improve the ranking of a state of the art IE system, TEXTRUNNER. For these experiments, we evaluate REALM against the TEXTRUNNER frequency-based ordering, a pattern-learning approach, and the HMM-T and REL-GRAMS components taken in isolation. The TEXTRUNNER frequency-based ordering

	`Conquered`	`Founded`	`Headquartered`	`Merged`	Average
Avg. Prec.	0.698	0.578	0.400	0.742	0.605
TextRunner	0.738	0.699	0.710	0.784	0.733
Pl	0.885	0.633	0.651	0.852	0.785
Pl+ Hmm-T	0.883	0.722	0.727	0.900	0.808
Hmm-T	0.830	0.776	0.678	0.864	0.787
Rel-grams	**0.929 (39%)**	0.713	0.758	0.886	0.822
Realm	0.907 **(19%)**	**0.781 (27%)**	**0.810 (35%)**	**0.908 (38%)**	**0.851 (39%)**

Table 4.3: Performance of Realm for assessment of sparse extractions. Listed is area under the precision/recall curve for each method. In parentheses is the percentage reduction in error over the strongest baseline method (TextRunner or Pl) for each relation. "Avg. Prec." denotes the fraction of correct examples in the test set for each relation. Realm outperforms its Rel-grams and Hmm-T components taken in isolation, as well as the TextRunner and Pl systems from previous work.

ranks extractions in decreasing order of their extraction frequency, and importantly, for our task this ordering is essentially equivalent to that produced by the Urns model or Pointwise Mutual Information methods described in Chapter 3.

The pattern-learning approach, denoted as Pl, is modeled after Snowball [2]. The algorithm and parameter settings for Pl were those manually tuned for the `Headquartered` relation in previous work [1]. A sensitivity analysis of these parameters indicated that the results are sensitive to the parameter settings. However, we found no parameter settings that performed significantly better, and many settings performed significantly worse. As such, we believe our results reasonably reflect the performance of a pattern learning system on this task. Because Pl performs relation assessment, we also attempted combining Pl with Hmm-T in a hybrid method (Pl+ Hmm-T) analogous to Realm.

The results of these experiments are shown in Table 4.3. Realm outperforms the TextRunner and Pl baselines for all relations, and reduces the missing area under the curve by an average of 39% relative to the strongest baseline. The performance differences between Realm and TextRunner are statistically significant for all relations, as are differences between Realm and Pl for all relations except `Conquered` ($p < 0.01$, two-sampled t-test). The hybrid Realm system also outperforms each of its components in isolation.

4.4 Related Work

To our knowledge, Realm is the first system to use language modeling techniques for IE Assessment.

Redundancy-based approaches to pattern-based IE assessment [20, 24] require that extractions appear relatively frequently with a limited set of patterns. In contrast, Realm utilizes *all* contexts to build a model of extractions, rather than a limited set of patterns. Our experiments demonstrate that Realm outperforms these approaches on sparse data.

Type checking using *named-entity taggers* has been previously shown to improve the precision of pattern-based IE systems [1, 27], but the Hmm-T type-checking component we develop differs from this work in important ways. Named-entity taggers are limited in that they typically recognize only small set of types (*e.g.*, ORGANIZATION, LOCATION, PERSON), and they require hand-tagged training data for each type. Hmm-T, by contrast, performs type checking for *any* type. Finally, Hmm-T does not require hand-tagged training data.

Pattern learning is a common technique for extracting and assessing sparse data (*e.g.* [1, 49, 44]). Our experiments demonstrate that Realm outperforms a pattern learning system closely modeled after [1]. Realm is inspired by pattern learning techniques (in particular, both use the distributional hypothesis to assess sparse data) but is distinct in important ways. Pattern learning techniques require substantial processing of the corpus after the relations they assess have been specified. Because of this, pattern learning systems are unsuited to Open IE. Unlike these techniques, Realm pre-computes language models which allow it to assess extractions for arbitrary relations at run-time. In essence, pattern-learning methods run in time linear in the number of relations whereas Realm's run time is constant in the number of relations. Thus, Realm scales readily to large numbers of relations whereas pattern-learning methods do not.

A second distinction of Realm is that its type checker, unlike the named entity taggers employed in pattern learning systems (*e.g.*, Snowball), can be used to identify arbitrary types. A final distinction is that the language models Realm employs require fewer parameters and heuristics than pattern learning techniques.

Similar distinctions exist between Realm and a recent system designed to assess sparse extractions by bootstrapping a *classifier* for each target relation [27]. As in pattern learning, constructing the classifiers requires substantial processing after the target relations have been specified, and a set of hand-tagged examples per relation, making it unsuitable for Open IE.

4.5 Future Work

The language modeling techniques used by Mfa (Hidden Markov Models and n-gram models) are basic and represent only a first step in utilizing language modeling for information extraction. Experiments with more sophisticated techniques that exploit additional language elements, such as character-level or document-level features (e.g., HMM-LDA [31]) and the recursive structure of language (e.g., probabilistic context-free grammars), is a key item of future work. Likewise, other methods for generalizing from sparse contextual information (e.g., performing Latent Semantic Analysis over context vectors) should be compared experimentally with the Hmm-T approach. Further, Web-specific structures found in HTML are highly indicative of particular unary types (for example, HTML wrappers have been shown to be extremely effective at identifying unary extractions [25, 58]). Developing language modeling techniques that utilize this additional structure is an exciting future direction.

One specific enhancement to Hmm-T we would like to explore involves handling terms with multiple meanings, which are widespread in information extraction and are particularly challenging for Hmm-T to assess. We discuss this direction in detail below.

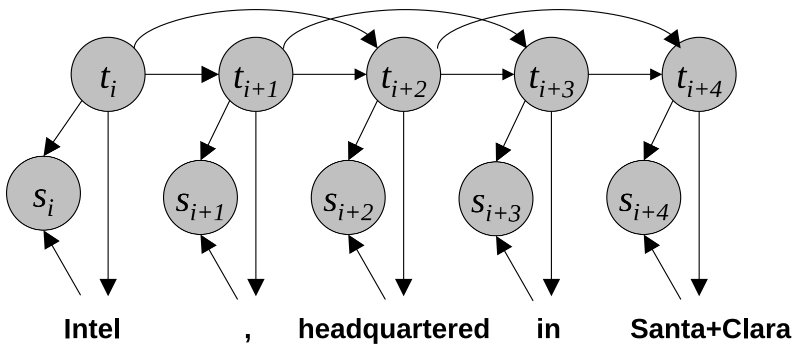

Figure 4.3: Graphical model employed by Pmm. The model includes sense variables s_i and state variables t_i for each token in the corpus.

4.5.1 Handling Polysemy

The Hmm-T language model faces a difficult problem with *polysemy*, the existence of terms that refer to multiple real-world entities. Consider the term `Chicago`, which most frequently refers to a city, but can also refer to a film. Hmm-T necessarily conflates all senses of each word w into a single distribution $P(t|w)$. As a result, the distributional similarity computation in Equation 4.2 is dominated by each term's majority sense, meaning that Hmm-T will not correctly type check terms for their minority senses (*e.g.*, the film sense of `Chicago`). In this section, we describe an enhancement to Hmm-T aimed at overcoming this limitation.

Our enhancement utilizes what we term a *Polysemous Markov Model* (PMM), which augments a standard HMM with an explicit *sense variable* for each observed token. A portion of the Bayes Net corresponding to the PMM is shown graphically in Figure 4.3. Formally, each token w_i in the corpus $\mathbf{w}$ is generated by a single hidden state t_i, and the state and token jointly generate a hidden *sense variable* s_i. The hidden states take integral values from $\{1, \ldots, T\}$ (as in Hmm-T), and the hidden senses take values from $\{1, \ldots, S_w\}$, where in general the number of senses S_w can vary across different terms w. The joint distribution of senses $\mathbf{s}$ and observed tokens $\mathbf{w}$ given the hidden states $\mathbf{t}$ is:

$$P(\mathbf{w}, \mathbf{s}|\mathbf{t}) = \prod_i P(s_i|t_i, w_i) P(w_i|t_i) P(t_i|t_{i-1}, \ldots, t_{i-k}) \tag{4.5}$$

The distributions on the right-hand-side of Equation 4.5 can be learned from a corpus in an unsupervised manner using Expectation-Maximization (EM). Inference complexity for PMMs is tractable, requiring a manageable factor of S_w more computations than HMMs on average.[6]

An important modeling decision lies in choosing the number of word senses S_w. The number of distinct senses for each term can vary wildly across a lexicon. A straightforward approach is to choose a fixed number of senses for each term, using for example the property (due to Zipf) that the number of meanings of a term varies with the square root of its frequency [37]. A more flexible, albeit less efficient alternative would be to employ a non-parametric technique (e.g., Dirichlet process mixture models [4]) to dynamically choose the number of senses for each term.

Unlike Hmm-T, Pmm can be utilized to determine if two entities share *any* sense in common, including minority senses. Given entities e_1 and e_2, we define a distance function (in terms of the parameters of a learned PMM) as:

$$d(e_1, e_2) = \min_{s,s'} KL(P(t|s, e_1), P(t|s', e_2)) \tag{4.6}$$

Thus, if e_1 has any sense which is similar to some sense of e_2, the entities are judged to be similar, even when the senses are minority senses.

The Pmm model gives a distribution over senses for each word occurrence in a corpus. Thus, Pmm is valuable for word sense disambiguation (WSD), the task of determining which sense of a term is being used in a particular sentence.

Because the PMM represents senses explicitly, it can be augmented to exploit the powerful heuristic that often, multiple uses of a word in a given document correspond to the same word sense [59]. One way to implement this in a PMM would involve generating all sentences in a document jointly, with sense variables interconnected by additional pairwise terms $P(s_i|s_j, t_i = t_j)$ favoring sense distributions in which s_i is distributed similarly to s_j whenever $t_i = t_j$. Of course, such an enhancement dramatically increases the interconnectivity of the graphical model. To maintain tractability, a form of approximate inference

[6]Tractability in parameter learning is further aided by the fact that the EM procedure can be parallelized in a straightforward fashion.

(*e.g.*, Gibbs sampling) can be employed to replace or augment the exact inference used in the original PMM.

4.6 Conclusions

This chapter demonstrated that unsupervised language models, as embodied in the REALM system, are an effective means of assessing sparse extractions.

Another attractive feature of REALM is its scalability. Scalability is a particularly important concern for *Open Information Extraction*, the task of extracting large numbers of relations that are not specified in advance. Because HMM-T and REL-GRAMS both precompute language models, REALM can be queried efficiently to perform IE Assessment. Further, the language models are constructed independently of the target relations, allowing REALM to perform IE Assessment even when relations are not specified in advance.

Chapter 5
CONCLUSION

In this thesis, we illustrated a probabilistic model of the KnowItAll Hypothesis, which states that extractions that occur more frequently in distinct sentences in a corpus are more likely to be correct. The model leverages the redundancy inherent in large text collections to identify correct extractions from the Web automatically, without the use of hand-labeled training data.

We began by presenting the KnowItAll Hypothesis as an instance of a general problem structure inherent in many textual classification tasks. The Monotonic Feature Model (MFM) holds for a classification task when one or more feature values exhibit a monotonic relationship with class probability. We showed theoretically that the MFM provided a number of valuable properties for learning from unlabeled data. In particular, when the identity of a single monotonic feature is known, PAC-learning is possible from unlabeled data alone. Further, the MFM is formally distinct from structures used in previous semi-supervised learning research, and offers greater information gain than labeled examples as the feature space increases in size, under certain assumptions.

We demonstrated that the MFM can be employed effectively in practice. In experiments with the "20 Newsgroups" document classification task, a learner given an MF and unlabeled data achieved accuracy equal to that of a state-of-the-art semi-supervised learner relying on 160 hand-labeled examples. Even when MFs are not given as input, a semi-supervised learning method that discovers MFs from small amounts of labeled data was found to reduce error by 15% on the newsgroups task.

While the MFM is effective in some cases, it has important limitations. We demonstrated theoretically and empirically that for more complex problems, including Unsupervised Information Extraction (UIE), the MFM is not sufficient. We then presented the Urns model, a special case of the MFM that addresses its limitations. Urns takes the form of a combinatorial balls-and-urns model, and computes the probability an extraction is a member of the target class based on its MF value. We proved theoretically that the Urns model can provide PAC learnability for problems in which the MFM alone is ineffective. The model was also effective in practice. In experiments with UIE we showed that Urns produces probability estimates that are on average 15 to 20 times closer to ideal, when compared with techniques from previous work. Further, without labeled data Urns provides performance equivalent to that of state-of-the-art semi-supervised approaches trained using over 320 labeled examples.

Because extraction frequency follows a Zipf distribution, even on the Web a substantial fraction of extractions appear infrequently. For these sparse extractions, Urns and other techniques that depend on the KnowItAll hypothesis alone are ineffective, because each extraction appears only a handful of times. We introduced the insight that the sub-field of statistical language modeling provides unsupervised methods that can be leveraged to assess sparse extractions, through the combination of the KnowItAll hypothesis and the Distributional hypothesis of language. In experiments in UIE, an assessment technique based on language models reduced extraction error by 39%, on average, when compared with techniques from previous work. Further, because unsupervised language models can be pre-computed without any labeled data or seed examples, they are ideally suited to UIE and are substantially more scalable than techniques from previous work.

5.0.1 Limitations and Future Work

The methods and theoretical results we presented have a number of limitations which could be addressed in future work. Many of the theoretical results rely on strong assumptions. In order to establish PAC-learnability in the MFM, we assume that the monotonic feature values are conditionally independent of the other feature values. While this assumption is not uncommon, and is employed in the co-training theory which our work builds upon, it is typically wildly untrue in practice. As is the case with co-training and Naive Bayes classifiers, however, MFM-based techniques which in theory require conditional independence are often effective in practice even when the assumption is violated. Theoretical results that characterize more precisely when the MFM is likely to be effective are an item of future work.

In cases in which the MFM is *not* effective, we proved that the Urns model is sufficient to provide PAC classification. The Urns model relies on its own set of idealizations. First, we assume that the target and error elements are Zipf distributed. Second, for the PAC-learnability result, we also require that as the data set grows an arbitrary number of unique target elements have frequencies above some reference frequency (along with a similar criteria for error elements). The second assumption is unlikely to hold even as the Web grows. Weakening this assumption is an item of future work. As with the MFM, however, we demonstrated that techniques based on the Urns model provide empirical benefits, despite unrealistic assumptions in the theory.

When utilizing Urns for UIE in practice, the EM-based algorithm we employ to learn Urns parameters from unlabeled data could be improved in a number of ways. The algorithm often requires a sample size of hundreds or thousands of unlabeled observations of each class in order to be effective (as illustrated in Figure 3.3). For classes where data is less plentiful, such as many of the relations extracted by the TextRunner system, the parameter learning algorithm is less effective. We expect that Urns could be modified to learn accurate parameters for much smaller data sets, through the use of priors or more robust likelihood-maximization techniques. These experiments are an area of future work.

Urns also requires that a reasonable estimate of the precision of the extraction process be known. We demonstrated that this requirement is not prohibitive when extracting instances of classes drawn from WordNet, using generic extraction patterns; the extraction frequency can be assumed or adjusted from unlabeled text in such a way that the probabilities produced by Urns still offer large improvements over previous techniques. However, for Open IE systems such as TextRunner which discover target relations from text, the situation is more complex. In TextRunner, extraction precision can vary greatly across the discovered relations; thus, the probabilities output by Urns in this case are less accurate. Methods for estimating extraction precision across relations in Open IE systems is an item of future work.

There are ways in which the accuracy of Urns could be improved, primarily by incorporating evidence from textual data other than a small set of generic patterns. The multiple urns model combines evidence across different extraction mechanisms, but the mechanisms must be pre-defined and conform to a stringent correlation model. The Realm system, on the other hand, incorporates the entire corpus of textual data when assessing extractions, and uses a much more flexible correlation model. However, Realm outputs only a ranking, rather than probabilities or classifications. While research-prototype IE systems are often evaluated in terms of how well they rank extractions, in practice a useful IE system requires the ability to classify extractions as correct or incorrect. Methods for generalizing the multi-urn model to incorporate more diverse textual data, or alternatively generalizing Realm to produce classifications or probability estimates, is an important direction for future work.

Our estimation of the highest-priority piece of future work is additional investigation into language modeling techniques for information extraction. Realm illustrates that statistical language modeling techniques provide accuracy and scalability improvements over previous techniques. Nonetheless, there is still substantial room for improvement as our experiments demonstrate. The language models employed by Realm are relatively simple, and ignore potentially relevant information including document characteristics, character-level features and the recursive structure of language syntax. It seems reasonable to believe that incorporating this additional information would improve the accuracy of language models for information extraction.

Lastly, as stated in the introduction a central goal of UIE is to enable radically improved search engines that can answer complex queries by synthesizing information across documents. At first glance, it certainly appears that logical inference will be required to perform this task in general— for example, by concluding from the statements "anti-oxidants prevent cancer" and "oranges contain anti-oxidants" the relation `Prevents(Oranges, Cancer)`, even though text directly supporting this extraction (e.g., "oranges prevent cancer") might never occur in the corpus [54]. Alternatively, however, we might ask whether a strong enough statistical language model could simply estimate the probability of the phrase "oranges prevent cancer," with which we could (via the KnowItAll Hypothesis) arrive at an accurate probability estimate for the associated extraction. Certainly, because this phrase follows logically from other phrases in the corpus, we would expect a language model of sufficient accuracy to assign the phrase a probability much larger than chance. Compared with performing true logical inference, the language modeling approach has the advantage of side-stepping a number of complex issues in knowledge representation and inference over large, noisy knowledge bases. However, constructing a language model of sufficient accuracy for this task presents challenges of its own. Further investigation of the potential of unsupervised statistical language models for performing UIE and improving Web search engines is a key item of future work.

BIBLIOGRAPHY

[1] E. Agichtein. *Extracting Relations From Large Text Collections.* PhD thesis, Department of Computer Science, Columbia University, 2005.

[2] E. Agichtein. Confidence estimation methods for partially supervised relation extraction. In *SDM 2006*, 2006.

[3] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *Procs. of the Fifth ACM International Conference on Digital Libraries*, 2000.

[4] Charles E. Antoniak. Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174, 1974.

[5] M. Banko, M. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the Web. In *Procs. of IJCAI*, 2007.

[6] M. Banko and O. Etzioni. The tradeoffs between traditional and open relation extraction. In *Proceedings of ACL*, 2008.

[7] R. Bellman. *Adaptive Control Processes: A Guided Tour.* Princeton University Press, 1961.

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

[9] M.E. Califf and R.J. Mooney. Relational Learning of Pattern-Match Rules for Information Extraction. In *Working Notes of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*, pages 6–11, Menlo Park, CA, 1998. AAAI Press.

[10] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001.

[11] Ming-Wei Chang, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar. Importance of semantic representation: Dataless classification. In Dieter Fox and Carla P. Gomes, editors, *AAAI*, pages 830–835. AAAI Press, 2008.

[12] Jinxiu Chen, Dong-Hong Ji, Chew Lim Tan, and Zheng-Yu Niu. Semi-supervised relation extraction with label propagation. In *HLT-NAACL*, 2006.

[13] F. Ciravegna. Adaptive Information Extraction from Text by Rule Induction and Generalisation. In *Procs. of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 1251–1256, Seattle, Washington, 2001.

[14] M. Collins and Y. Singer. Unsupervised Models for Named Entity Classification. In *Procs. of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–111, Maryland, USA, 1999.

[15] F. Crestani, M. Lalmas, C. Rijsbergen, and I. Campbell. Is this document relevant?...probably: A survey of probabilistic models in information retrieval. *ACM Computing Surveys*, 30(4), 1998.

[16] A. Culotta and A. McCallum. Confidence estimation for information extraction. In *HLT-NAACL*, 2004.

[17] Marie-Catherine de Marneffe, Anna Rafferty, and Christopher D. Manning. Finding contradictions in text. In *ACL 2008*, 2008.

[18] D. Downey, M. Broadhead, and O. Etzioni. Locating complex named entities in web text. In *Procs. of IJCAI*, 2007.

[19] D. Downey and O. Etzioni. Look ma, no hands: Analyzing the monotonic feature abstraction for text classification. In *Advances in Neural Information Processing Systems (NIPS) 21, 2009*, January 2009.

[20] D. Downey, O. Etzioni, and S. Soderland. A Probabilistic Model of Redundancy in Information Extraction. In *Procs. of IJCAI*, 2005.

[21] D. Downey, S. Schoenmackers, and O. Etzioni. Sparse information extraction: Unsupervised language models to the rescue. In *Proc. of ACL*, 2007.

[22] Gregory Druck, Gideon Mann, and Andrew McCallum. Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 2008.

[23] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Web-Scale Information Extraction in KnowItAll. In *WWW*, pages 100–110, New York City, New York, 2004.

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

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

[26] O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Methods for domain-independent information extraction from the Web: An experimental comparison. In *Procs. of the 19th National Conference on Artificial Intelligence (AAAI-04)*, pages 391–398, San Jose, California, 2004.

[27] R. Feldman, B. Rosenfeld, S. Soderland, and O. Etzioni. Self-supervised relation extraction from the Web. In *International Symposium on Methodologies for Intelligent Systems*, pages 755–764, 2006.

[28] D. Freitag and A. McCallum. Information Extraction with HMMs and Shrinkage. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*, Orlando, Florida, 1999.

[29] W. A. Gale and G. Sampson. Good-turing frequency estimation without tears. *Journal of Quantitative Linguistics*, 2(3):217–237, 1995.

[30] Alfio Gliozzo, Carlo Strapparava, and Ido Dagan. Investigating unsupervised learning for text categorization bootstrapping. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 129–136, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[31] Thomas L. Griffiths, Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. Integrating topics and syntax. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 537–544. MIT Press, Cambridge, MA, 2005.

[32] Z. Harris. Distributional structure. In J. J. Katz, editor, *The Philosophy of Linguistics*, pages 26–47. New York: Oxford University Press, 1985.

[33] M. Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Procs. of the 14th International Conference on Computational Linguistics*, pages 539–545, Nantes, France, 1992.

[34] Thorsten Joachims. *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms.* Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[35] W. Lin, R. Yangarber, and R. Grishman. Bootstrapped Learning of Semantic Classes from Positive and Negative Examples. In *Procs. of ICML-2003 Workshop on The Continuum from Labeled to Unlabeled Data*, pages 103–111, Washington, D.C, 2003.

[36] Wei-Liem Loh. Estimating the mixing density of a mixture of power series distributions. In S. S. Gupta and J. O. Berger, editors, *Statist. Decision Theory and Related Topics V*, pages 87–98. Springer, New York, 1993.

[37] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing.* 1999.

[38] A. McCallum. Efficiently Inducing Features of Conditional Random Fields. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 403–410, Acapulco, Mexico, 2003.

[39] Rada Mihalcea and Dan I. Moldovan. An automatic method for generating sense tagged corpora. In *AAAI/IAAI*, pages 461–466, 1999.

[40] Brian Milch, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. Blog: Probabilistic models with unknown objects. In Luc De Raedt, Thomas Dietterich, Lise Getoor, and Stephen H. Muggleton, editors, *Probabilistic, Logical and Relational Learning - Towards a Synthesis*, number 05051 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2006. <http://drops.dagstuhl.de/opus/volltexte/2006/416> [date of citation: 2006-01-01].

[41] Tom M. Mitchell. *Machine Learning.* McGraw-Hill, New York, 1997.

[42] K. Nigam, J. Lafferty, and A. McCallum. Using Maximum Entropy for Text Classification. In *Procs. of IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, Stockholm, Sweden, 1999.

[43] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[44] M. Paşca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. Names and similarities on the web: Fact extraction in the fast lane. In *Procs. of ACL/COLING 2006*, 2006.

[45] Marius Pasca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. Organizing and searching the world wide web of facts - step one: The one-million fact extraction challenge. In *AAAI 2006.* AAAI Press, 2006.

[46] L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[47] D. Ravichandran, P. Pantel, and E. H. Hovy. Randomized Algorithms and NLP: Using Locality Sensitive Hash Functions for High Speed Noun Clustering. In *Procs. of ACL 2005*, 2005.

[48] M. Richardson and P. Domingos. Markov Logic Networks. *Machine Learning*, 62(1-2):107–136, 2006.

[49] E. Riloff and R. Jones. Learning Dictionaries for Information Extraction by Multi-level Boot-strapping. In *Procs. of AAAI-99*, pages 1044–1049, 1999.

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

[51] A. Ritter, S. Soderland, D. Downey, and O. Etzioni. It's a contradiction–no, it's not: A case study using functional relations. In *EMNLP*, 2008.

[52] S. E. Robertson, S. Walker, M. Hancock-Beaulieu, A. Gull, and M. Lau. Okapi at TREC-3. In *Text REtrieval Conference*, pages 21–30, 1992.

[53] Robert E. Schapire. The strength of weak learnability. *Mach. Learn.*, 5(2):197–227, 1990.

[54] S. Schoenmackers, O. Etzioni, and D. Weld. Scaling Textual Inference to the Web. In *Procs. of EMNLP*, 2008.

[55] M. Skounakis and M. Craven. Evidence combination in biomedical natural-language processing. In *BIOKDD*, 2003.

[56] S. Soderland. Learning Information Extraction Rules for Semi-Structured and Free Text. *Machine Learning*, 34(1-3):233–272, 1999.

[57] P. D. Turney. Mining the Web for Synonyms: PMI–IR versus LSA on TOEFL. *Lecture Notes in Computer Science*, 2167:491–502, 2001.

[58] Richard C. Wang and William W. Cohen. Language-independent set expansion of named entities using the web. In *ICDM*, pages 342–350, 2007.

[59] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Meeting of the Association for Computational Linguistics*, pages 189–196, 1995.

[60] A. Yates and O. Etzioni. Unsupervised resolution of objects and relations on the Web. In *Procs. of HLT*, 2007.

Appendix A
DOWNLOADABLE RESOURCES

Most of the data sets and code bases utilized in this thesis are available for download at http://turing.cs.washington.edu/papers/downey_thesis_data/.

Included are the following data sets:

- **Chapter 2**: The ground truth, unlabeled examples, and original corpus used for evaluation in the information extraction experiments.
- **Chapter 3**: The ground truth used in the supervised and unsupervised experiments, along with the measurements of the precision of extractions from various classes (see Section 3.4.4.
- **Chapter 4**: The ground truth and corpus used for all experiments.

Also included are the following models and code bases:

- **Chapter 3**: Code for learning urns parameters from a set of MF values, and computing probabilities using the parameters.
- **Chapter 4**: An HMM-T model of latent state probabilities given terms, $P(t|w)$, trained on the corpus used for evaluation.[1]

[1]Due to a hard drive failure, the particular HMM-T model evaluated in the type checking experiments in Chapter 4 was lost. However, the model available for download performs slightly better on average than the type checking results reported in Table 4.1 (0.877 average AUC, vs. 0.849).

VITA

Doug Downey received a B.S. and M.S. degree in Computer Science from Case Western Reserve University in 2000. Starting in 2002, he attended the University of Washington, where he was advised by Oren Etzioni. He was supported by a National Science Foundation fellowship, and a Microsoft Research Graduate Fellowship sponsored by Microsoft Live Labs. He received an M.S. degree in 2004, and a Ph.D. in 2008, both in Computer Science from the University of Washington.