Constrained Semi-supervised Learning in the Presence of Unanticipated Classes

Bhavana Bharat Dalvi
bbd@cs.cmu.edu

CMU-LTI-15-006

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:
Prof. William W. Cohen (Co-Chair), Carnegie Mellon University
Prof. Jamie Callan (Co-Chair), Carnegie Mellon University
Prof. Tom Mitchell, Carnegie Mellon University
Dr. Alon Halevy, Google Research

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies.

Keywords: Semi-supervised learning, Clustering, New class discovery, Knowledge bases, Ontologies, Entities on the Web, Multi-view Learning

Dedicated to my beloved parents Mr. Bharat Madhav Dalvi and Mrs. Bhakti Bharat Dalvi.

Acknowledgments

The process of PhD is challenging in many ways, and it would have been impossible to survive without invaluable support from everyone around me. This is a heartfelt attempt to express my gratitude to everyone who helped me during this journey.

First and foremost, I would like to thank my advisors Prof. William W. Cohen and Prof. Jamie Callan for their extensive support, and patient guidance over past six years. They always listened to my ideas carefully, however crazy they were. I have learned a lot from them over these years. Prof. Cohen set an example for all of us by being an awesome researcher, advisor, teacher, and a kind person. I like his way of putting research projects in broad context. His encouraging words and optimism made me feel good about myself during difficult times in PhD. He was always approachable whenever I needed any guidance. I will always keep in mind the way he perfectly manages his professional, and personal life. I am equally thankful to Prof. Callan for co-advising my thesis. His questions during weekly meetings always challenged me and helped me keep on track and discover mistakes quickly. He has taught me to think deeply about research problems, and to always meaningfully question and challenge one's own work. I deeply admire his presentation and teaching skills.

I would also like to thank my other thesis committee members Prof. Tom Mitchell, Dr. Alon Halevy. I am grateful to Prof. Tom Mitchell for his valuable comments on my research during NELL group meetings. I am amazed at his breadth of knowledge and enthusiasm. The brainstorming sessions headed by Prof. Mitchell have helped me a lot in coming up with novel ideas. I sincerely thank Dr. Alon Halevy for readily agreeing to serve as the external member of my thesis committee, for diligently reading my drafts), and providing feedback. I am thankful to Dr. Halevy and Dr. Anish Das Sarma for their mentorship during my internship at Google Research.

My sincere thanks to my family members especially my parents, husband (Aditya Mishra), and in laws. I am eternally grateful to my parents for giving me unconditional love and support throughout my life. Their continuous care and love made me survive during the thick and thin. They always put my interests above theirs and imbibed in me the importance of education. Aditya, has been beside me during this long journey through grad school. He patiently listened to my ideas, questions, and concerns and always gave me most pertinent advice. He was always available whenever I needed him. For that, no amount of gratitude can suffice. His optimism, relentless support and love keeps me going. I am equally grateful to my in laws for encouraging me to continue my research work during the post marriage years. They always lifted my spirits which helped me give my best at research.

I would also like to thank my friends Anjali, Archna, Athula, Derry, Kriti, Meghana, Naman, Pallavi, Prasanna, Ramnath, Ranjani, Reyyan, Rose, Ruta, Sunayana, Siddharth and Subhodeep for their constant support during these years. They provided a stimulating environment making my graduate student life enjoyable and truly memorable. I would also like to thank members of the NELL group and Prof. Callan's group for the countless thought provoking conversations during the group meetings. Finally my heartfelt thanks to Stacey Young, Sharon Cavlovich, Sandra Winkler, and the rest of the staff at CMU for their invaluable help and assistance over the years.

Abstract

Traditional semi-supervised learning (SSL) techniques consider the missing labels of unlabeled datapoints as latent/unobserved variables, and model these variables, and the parameters of the model, using techniques like Expectation Maximization (EM). Such semi-supervised learning techniques are widely used for Automatic Knowledge Base Construction (AKBC) tasks.

We consider two extensions to traditional SSL methods which make it more suitable for a variety of AKBC tasks. First, we consider jointly assigning multiple labels to each instance, with a flexible scheme for encoding constraints between assigned labels: this makes it possible, for instance, to assign labels at multiple levels from a hierarchy. Second, we account for another type of latent variable, in the form of unobserved *classes*. In open-domain web-scale information extraction problems, it is an unrealistic assumption that the class ontology or topic hierarchy we are using is complete. Our proposed framework combines structural search for the best class hierarchy with SSL, reducing the semantic drift associated with erroneously grouping unanticipated classes with expected classes. Together, these extensions allow a single framework to handle a large number of knowledge extraction tasks, including macro-reading, noun-phrase classification, word sense disambiguation, alignment of KBs to wikipedia or on-line glossaries, and ontology extension.

To summarize, this thesis argues that many AKBC tasks which have previously been addressed separately can be viewed as instances of single abstract problem: multiview semi-supervised learning with an incomplete class hierarchy. In this thesis we present a generic EM framework for solving this abstract task.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Extracting knowledge from the Web and integrating it into a coherent knowledge base (KB) is a task that spans the areas of natural language processing, information extraction, information integration, databases, search, and machine learning. Building large KBs automatically is important because they can benefit many other natural language processing tasks like entity linking, word sense disambiguation, question answering and web search. Recent years have seen significant advances in the area of Automatic Knowledge Base Construction (AKBC), with techniques ranging from unsupervised to supervised learning.

AKBC systems extract entities and relationships between them using clues like text context patterns. They employ techniques like classification, named entity disambiguation, coreference resolution to solve various subproblems in this area. NELL (Never Ending Language Learning) is an example AKBC system that uses semi-supervised learning (SSL) techniques. These SSL techniques take as input a hand curated ontology of around 275 classes, few seed examples of each class, and a large unlabeled dataset to produce more facts that can be added to the knowledge base.

With the availability of web scale corpora of semi-structured data in the form of HTML tables and unstructured data in the form of text, there is a need for developing information extraction techniques that will work with such datasets and help populate KBs. In Chapter 3 we propose an unsupervised clustering technique named WebSets [126] that works on HTML tables on the Web. WebSets runs a bottom-up clustering algorithm on entities extracted from these tables, and produces named and coherent sets of entities in a completely unsupervised fashion. WebSets discovers many coherent, high-quality concepts hidden in the unlabeled data which are not present in a large ontology like NELL. We also explored a semi-supervised version of WebSets that does constrained clustering using the facts from the NELL Knowledge Base (KB) as seeds.

The WebSets experiments revealed several less-studied problems associated with web-scale information extraction, and semi-supervised learning in general. It suggested the following research questions: 1) In practice, the information extraction system knows about some concepts, and some of their example instances, but unlabeled datasets contain many more concepts that are not known upfront. Can an algorithm do both semi-supervised learning for the classes with training data and unsupervised clustering for classes without training data? 2) Another important aspect of the knowledge extraction problem is that each noun phrase can have different sets of features e.g., features based on occurrences of noun phrases in text, HTML tables etc. Can the

existing semi-supervised learning methods be easily extended to combine scores of classifiers learnt from different data views? 3) The categories in a KB ontology are related to each other through subset and mutual exclusion constraints. Can a semi-supervised learning method leverage these class constraints to learn better classifiers? Here, we propose semi-supervised learning techniques that can tackle all three issues in a unified learning framework.

1.1 Thesis Contributions

Let us now expand on the above mentioned research questions. More specifically, this thesis focuses on extending semi-supervised learning techniques to incorporate 1) unanticipated classes, 2) multi-view data, and 3) ontological constraints. Next, we present a brief summary of contributions done in each of these three settings and some combinations of them.

1.1.1 Semi-supervised Learning in the Presence of Unanticipated Classes

Existing KBs contain very useful information about concepts present in the world along with some known instances of these concepts. Existing semi-supervised learning algorithms can make use of such information to acquire additional instances of the known concepts. However, we believe that there are many more concepts hidden in the unlabeled data. For example, if the information extraction system knows about four classes: mammals, reptiles, fruits and vegetables; the unlabeled data can contain instances of classes like birds, cities, beverages and so on. Similarly, in our WebSets experiments we found that there are coherent clusters like "police designation", "social metric", "government type" that are present in the unlabeled data but not present in the NELL ontology.

This problem is especially challenging when we have very small amount of training data for the known classes and there might be large number of unanticipated classes hidden in the unlabeled data. Our experiments showed that when the number of seed classes are much smaller than actual, more iterative semi-supervised learning methods like classification EM result in semantic drift for the seed classes.

To tackle this scenario, we combined the semi-supervised learning of known concepts with unsupervised discovery of new concepts, a method called "Exploratory Learning". This method is an "exploratory" extension of expectation-maximization (EM) that explores different numbers of classes while learning. The intuition we use is that a new class should be introduced to hold x when the probability of x belonging to existing classes is close to uniform. With this extension, the method becomes more robust to the presence of unanticipated classes present in the unlabeled data.

On various publicly available datasets we found that our proposed methods improved over traditional semi-supervised learning methods with 28% to 190% relative improvements in terms of seed class F1. We also show experimentally that our new class creation criterion is more effective when compared to a baseline that uses Chinese Restaurant Process in terms of number of classes produced, seed class F1 and runtime.

1.1.2 Semi-supervised Learning with Multi-view Data

In multiclass semi-supervised learning, sometimes the information about datapoints is present in multiple views. For example, a noun-phrase "Pittsburgh" can be represented by two data views: its occurrences along with text contexts and its occurrences in HTML table columns. Similarly, In the WebSets system, we can construct multi-view features by representing each noun-phrase by 1) a set of co-occurring noun-phrases, 2) a set of table columns it appears in, 3) a set of hypernyms it appears with in the context of Hearst patterns, and so on. All these clues can be leveraged to classify a noun-phrase more effectively.

Trivial solution of concatenating the feature vectors from multiple views need not be the best choice, and different models might suit well for different views. This multi-view learning task gets even more challenging when the amount of training data is small. Hence an intelligent way to combine scores of models built from multiple views is a promising direction. NELL uses an ad hoc multi-strategy approach, instead we focus on systematic evaluation of multi-view SSL.

In Chapter 5, we propose optimization based methods to tackle semi-supervised learning in the presence of multiple views. Our techniques make use of mixed integer linear programming formulations along with the EM framework to find consistent class assignments given the scores in each data view. We found that several existing and some novel score combination strategies can be represented in a single optimization framework, and it results in improvements over a simple baseline that concatenates feature vectors from different views.

We present extensive experiments on 8 different multi-view datasets, including document classification, image classification, publication categorization and knowledge based information extraction datasets. We showed that our techniques give state-of-the-art performance when compared to existing multi-view learning methods including the co-training based algorithm proposed by Bickel and Scheffer [10], on the problem of flat multi-view semi-supervised learning.

1.1.3 Semi-supervised Learning in the Presence of Ontological Constraints

Multiple views are only one issue arising in complex real-world learning tasks. For instance, in the above mentioned KB population task, labels assigned to each noun-phrase need to be consistent with the hierarchical class constraints posed by the KB ontology. Example class constraints include: "if an entity is classified as Mammal then it should also be classified as Animal" (subclass-superclass constraint), "if an entity is classified as Mammal then it should not be classified as Reptile" (mutual exclusion constraint). Further, the class hierarchy need not be a tree or some of the classes might be overlapping.

Here each datapoint is assigned a bit vector of labels, one bit per class in the ontology. The ontological constraints tell us which bit vectors are consistent and which are invalid. Mixed integer linear programming formulations are used to estimate optimal label vectors in each iteration of EM algorithm.

We used this constrained SSL method, named GLOFIN, for the task of automatically finding glosses for entities in a gloss-free knowledge base like NELL (presented in Section 6.3.2). For example, "Microsoft is a software company headquartered in Redmond ..." can be assigned as a gloss for entity "Microsoft" of type "Company" in the NELL KB. GLOFIN classifies each candidate gloss into a set of KB categories while following ontological constraints which in turn

enables us to assign the gloss to the correct entity in the KB. In our gloss finding experiments we observe that the GLOFIN method outperforms the widely used SVM and label propagation baselines especially with small amount of noisy seed data.

1.1.4 Combinations of Scenarios

Thus we propose systematic solutions for incorporating unanticipated classes, multi-view data, and ontological constraints in the semi-supervised learning setting. One can think of various combinations of these scenarios. In this thesis we present results on two such combinations 1) multiple data views, ontological constraints and 2) unanticipated classes, ontological constraints.

Hierarchical Multi-view Semi-supervised Learning

In Section 6.2 we present semi-supervised learning techniques that can incorporate ontological class constraints as well as multiple data views. Here we used a mixed integer linear programming formulation that contains consistency constraints w.r.t. both class ontology and multiple data views.

On NELL's entity classification dataset we observed that this combination was better than the flat multi-view variant of our method. Further our method produced better results than co-training based algorithm proposed by Bickel and Scheffer [10] and simpler score aggregation methods like summation and multiplication of scores especially when the amount of labeled data is very small.

Hierarchical Exploratory Learning

Finally, the problem becomes even more challenging when the given class ontology is not complete to represent the unlabeled data. In Chapter 7 we present hierarchical exploratory learning, that can do semi-supervised learning in the presence of an incomplete class ontology. Our proposed method traverses the class hierarchy in top-down fashion to detect whether and where to add a new class for the given datapoint. It also uses a systematic optimization strategy to find the best set of labels for a datapoint given ontological constraints in the form of subset and mutual exclusion constraints.

On NELL's entity classification datasets this method outperforms both previously proposed flat exploratory learning method and its naive extension using divide and conquer strategy in terms of seed class F1 on average by 10% and 7% respectively.

1.2 Thesis Statement

In summary, our thesis statement is described as follows: Performance on a wide range of AKBC tasks can be improved by enhancing SSL methods to support 1) unanticipated classes, 2) ontological constraints between categories, and 3) multiple data views.

1.3 Additional Potential Applications

Automatic knowledge base construction (AKBC) spans the areas of natural language processing, information extraction, information integration, databases, search and machine learning. A variety of such AKBC tasks are being studied by the research community, including but not limited to entity classification, relation extraction, entity linking, slot filling, ontology alignment, and so on.

For most of these tasks the techniques involve a multiple step approach. An important step is to classify or cluster instances into one of the KB categories or relations. This inference task gets challenging when the amount of labeled data is very small. Semi-supervised or weakly supervised learning methods have been proposed to make effective use of available supervision (small amount of seed data or distant supervision) and combine it with large amount of unlabeled data, to learn models that will work for web-scale information extraction tasks.

In this thesis, we develop semi-supervised learning techniques in the presence of unanticipated classes, multiple data views, ontological class constraints, and combinations of these three scenarios. Below we present a list of actual and potential applications to indicate the potential scope of the methods presented in this thesis.

- **Macro-reading:** This task refers to semi-supervised classification of noun-phrases into a big taxonomy of classes, using distributional representation of noun-phrases. For example, clasifying a noun-phrase "Pittsburgh" by considering all text contexts it appeared with is a macro-reading task. NELL [25] is an example of macro-reading systems. We proposed an exploratory learning method [128] for macro-reading that reduces the semantic drift of seeded classes hence helping the traditional semi-supervised learning objectives.
- **Micro-reading:** Micro-reading differs from macro-reading in the sense that instead of using collective distributional features of a noun-phrase, we are trying to disambiguate an occurrence of noun-phrase w.r.t. the local context within a sentence or paragraph.

  We have applied our exploratory learning technique for clustering NIL entities (the entities that do not correspond to any of the existing entities in the KB) in the KBP entity discovery and linking (EDL) task [86]. We are working on using such hierarchical semi-supervised learning techniques for the task of word sense disambiguation by considering occurrences of all monosemous words as training data and polysemous word occurrences as unlabeled data. In these experiments, we use the WordNet synset hierarchy to deduce ontological constraints.
- **Multi-view macro-reading and micro-reading:** This task refers to what the "macro-reading" does, plus collecting signals from multiple data views. For example, NELL [25] proposed an information extraction system that classifies noun-phrases into a class hierarchy using clues from several different sources: text patterns occurring in sentences, morphological features, semi-structured data in the form of lists and tables etc. We proposed a multi-view semi-supervised learning method [123] for this task. An example of "multi-view micro reading" is a word sense disambiguation task where there are multiple data views of word synsets. For example, resources like WordNet, Wiktionary contain for each word sense a gloss and a set of example usages. One can use the glosses and example usages as multiple views to train multi-view models.

- **Alignment of online glossaries to an existing Knowledge Base:** This task can also be viewed as a gloss finding task for an existing gloss-free knowledge base. This task is important because a KB with glosses has been shown to be helpful for the task of entity recognition and disambiguation from search queries [26, 129].
- **Ontology extension:** Most of the concept or topic hierarchies available are incomplete to represent entities present on the Web. This task refers to the problem of discovering new classes that can be added to existing concept hierarchies to make them representative of the real world data. Many techniques have been proposed [92, 99, 116] with similar goals, however they are applicable in limited settings. We proposed a unified hierarchical exploratory learning technique OptDAC-ExploreEM that can populate known seeded classes in an ontology along with extending it with newly discovered concepts while taking care of ontological constraints [124, 127].

1.4 Thesis Outline

The rest of the document is organized as follows. In Chapter 2, we go through the background material for this thesis. Chapter 3 is a case study that presents our unsupervised information extraction technique WebSets [126] and its semi-supervised version. Chapter 4 dives into the novel problem of "Exploratory Learning" that makes the semi-supervised learning technique robust to the presence of unanticipated classes.

We then present our contributions in constrained semi-supervised learning. Multi-view semi-supervised learning is discussed in Chapter 5, followed by incorporation of ontological constraints in Chapter 6. Hierarchical exploratory learning is presented in Chapter 7 followed by conclusions and future research directions in Chapter 8.

Chapter 2

Background

This chapter gives us some background needed to understand the motivation behind this thesis research. Each of the later chapters go into greater details about related work relevant to that chapter. Here, we start with a broad summary of what is information extraction, followed by an introduction to one of its sub-fields, automatic knowledge base population. We then describe NELL, an example machine learning system that does KB population. Finally we present a list of AKBC problems that this thesis revolves around.

2.1 Information Extraction

Our techniques are motivated by applications of semi-supervised learning to variety of AKBC tasks. Knowledge base completion is a type of information extraction. Information extraction [111] refers to the area of research that focuses on the task of extracting structured information such as entities, relationships between entities, and attributes of entities from unstructured data like web text. With the availability of large amount of structured and unstructured data on the Web, there is a need for much richer forms of queries than mere keyword search. Information extraction enables intelligent querying, and analyzing of unstructured data by annotating it with the extracted information. However, extraction of useful structure from noisy sources of information on the Web is a challenging task. Many natural language processing (NLP) and machine learning (ML) techniques have been developed in the last two decades to solve this problem.

Let us start with an example of an information extraction task in the context of scholarly articles. If we have a large corpus of research publications, the naive way of organizing this information would be to build inverted indexes from each word in the corpus to all the papers it occurred in. This enables simple keyword based information retrieval. Information extraction techniques can enable us to extract structured information like author, title, conference name and citations from these research papers. Such structured information can enable new kinds of queries like "Find papers co-authored by two of the given authors", "Find most influential ten papers of a particular author" (a paper's influence is measured by how many papers cited this paper), and so on.

Information extraction has applications in various domains ranging from organization of domain specific information like scholarly articles, medical records etc., to building generic knowl-

edge bases that can represent general knowledge about the world e.g., Freebase, NELL, and YAGO. Output of information extraction systems is very useful for other real world applications like question answering, sentiment analysis, product recommendation, and so on.

2.2 Building Knowledge Bases

Information extraction in its earlier years focused on acquiring facts from an individual document in isolation. Such techniques learn a classifier for detecting a named entity of type Person/Location/Organization in specific documents. For example, while making a decision whether a noun-phrase "Pittsburgh" that occurred in document-8 at offset 32 is of type Location, we are looking at the text that appears only in document-8 and around offset 32. This kind of learning is also referred to as "micro-reading", where the decisions are made considering only small amount of context around an occurrence of a word or noun-phrase.

In the recent years, an orthogonal direction of research has emerged, which tries to gather information about an entity or a potential entity scattered among a large document collection. For example, while making a decision about whether a noun-phrase "Pittsburgh" is of type Location, we are looking at text contexts that appeared around this noun-phrase not only in document-8, but all documents in the corpus. This kind of learning is referred to as "macro-reading". It makes use of possibly redundant, complementary, or even conflicting pieces of information about a noun-phrase, integrates this information to do corpus level inference. Note that for an ambiguous noun-phrase like "Apple", a macro-reading system like ConceptResolver [74] learns that it can refer to either "Apple a Fruit" or "Apple a Company".

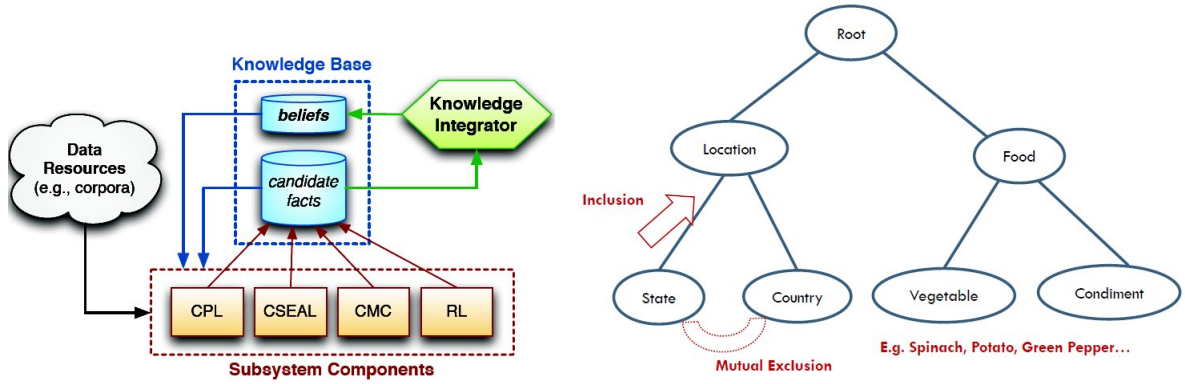
Facts extracted by information extraction systems can be stored in a database called a Knowledge Base (KB). Many popular KBs store the facts that the macro-reader extracted along with the confidence score for that fact. For example, a KB might store triples of the form: (Pittsburgh, Location, 0.99), indicating that the system is confident with a score of 0.99 that a noun-phrase "Pittsburgh" is of type "Location". Along with extracting type information, information extraction techniques are also capable of extracting relationships between entities. For example, it can learn which text patterns indicate whether a company is headquartered in a particular city and extract many such pairs from the corpus like (Microsoft, Redmond), (Google, Mountain View) and so on.

Traditionally knowledge bases were created and maintained by human annotators e.g., Freebase, WordNet etc. However, to build knowledge bases with high recall, such a manual process is expensive in terms of both time and monetary requirements. Hence in recent years, a lot of research efforts [64] are targeted to achieve Automatic Knowledge Base Construction (AKBC). The AKBC community is developing automated computer systems to process large amounts of unstructured and semi-structured information on the Web, to discover new facts about named entities and to add them to a KB. The information extraction problems studied in this thesis are more specifically subproblems of AKBC.

2.3 NELL: Never Ending Language Learning

The NELL project at CMU is an attempt to create a machine learning based computer system that can process information present on the Web and extract facts of the kind: 'Pittsburgh is of type City', 'CMU is of type University', and 'CMU is located in Pittsburgh'. Thus NELL is a specific AKBC system, and many of the tasks studied in this thesis are inspired by NELL as a usecase. This computer system is called Never-Ending Language Learner (NELL) because it is been running continuously since January 2010, attempting to extract facts from text found in hundreds of millions of web pages. Further it tries to improve its reading competence over iterations, so that the next iteration of the system can extract more facts from the web, and even more accurately. Currently, NELL has acquired around 2.4M high confidence facts from the Web.

Carlson et al. [24] described the architecture for the Never Ending Language Learning (NELL) system. The NELL system runs 24/7, reads textual and semi-structured data from the Web and populates a structured knowledge base by populating an existing ontology of classes and relations. It has two important goals. The first is to extract more facts, using models learned in previous iterations, and the second is to improve its reading competence so that the system gets better and better in extracting such knowledge.



(a) Architecture of NELL system (taken from Carlson et al. [24]).

(b) Example ontology.

Figure 2.1: NELL illustrations.

Figure 2.1 (a) shows the general architecture of the NELL system. In each iteration multiple information extraction (IE) systems like CPL, CSEAL [25] etc. propose candidate instances that could be added to the existing KB. These components supply a probability for each proposed candidate and a summary of the source evidence supporting it. The knowledge integrator promotes those beliefs to the KB for which multiple IE components have high score and agreement. Facts added in each iteration then contribute to making the extractors better in the next iteration.

Figure 2.1 (b) shows an example class ontology that could be used as input to NELL. There is an inclusion relationship between a class and its parent: e.g., if an entity is labeled as "Country" then it should also be labeled as "Location". Furthermore there are mutual-exclusion constraints between some class pairs: e.g., if an entity is labeled as "State", then it should not be labeled

as “Country”. There are seed examples for each of the concepts in the ontology: e.g., Spinach, Potato etc. are seed examples of the “Vegetable” class. Thus the NELL system takes as input an ontology of concepts that includes inclusion and mutual exclusion class constraints, along with seed examples, iteratively learns models for concepts, and populates them with new instances, without violating the class constraints.

Chapter 3

Case Study: Table Driven Information Extraction

In this chapter, we describe an open-domain information extraction method for extracting concept-instance pairs from an HTML corpus. This work is published in the proceedings of WSDM 2012 [126]. This chapter also illustrates the motivations behind the three semi-supervised learning scenarios that are focus of this thesis.

The most effective earlier approaches to the problem of concept-instance pair extraction (e.g., Pantel et al. [100], Van Durme and Pasca [132]) rely on combining clusters of distributionally similar terms and concept-instance pairs obtained with Hearst patterns[1][61]. In contrast, our method relies on a novel approach for clustering terms found in HTML tables, and then assigning concept names to these clusters using Hearst patterns. Note here that the names of concepts are not coming from a fixed ontology. The method can be efficiently applied to a large corpus, and experimental results on several datasets showed that our method can accurately extract large numbers of concept-instance pairs.

Our hypothesis is that HTML tables on the Web would contain lots of useful relational data. Let us look at some example tables that contain coherent sets of entities in their columns. Consider the example tables containing countries and their capitals shown in Figure 3.1. Notice that co-occurrences of entities in these tables can give us a strong signal that there are two sets of entities in these tables, one is 'Amsterdam, Andorra la Vella, Athens, Belgrade ...' (city names) and the other is 'Netherlands, Andorra, Greece, Serbia, ...' (country names). In this chapter we will describe a method that extracts such entity clusters and labels them in an unsupervised fashion.

3.1 WebSets: Unsupervised Concept-Instance Pair Extraction from HTML Tables

What can we do with a web scale corpus of HTML pages? Can we efficiently extract all possible concept-instance pairs from such corpus without any human input? What if we don't know how

[1]Hearst patterns are surface patterns (like "X such as Y") indicating that (X,Y) is a concept-instance pair.

Capital City	Satellite View and Map	Citizens	Country
Amsterdam The Hague (Den Haag; seat of govt)	Amsterdam Map The Hague	730,000	Netherlands
Andorra la Vella	Andorra la Vella Map	16,000	Andorra
Athens (Athína)	Athens Map	770,000	Greece
Belgrade (Beograd)	Belgrade Map	1,100,000	Serbia
Berlin	Berlin Map	3,400,000	Germany
Bern (Berne, Berna)	Bern Map	130,000	Switzerland
Bratislava	Bratislava Map	450,000	Slovakia
Brussels (Bruxelles, Brüssel, Brussel)	Brussels Map	140,000	Belgium
Bucharest (Bucuresti)	Bucharest Map	2,100,000	Romania
Budapest	Budapest Map	1,970,000	Hungary
Chisinau	Chisinau Map	670,000	Moldova
Copenhagen (København)	Copenhagen Map	1,360,000	Denmark
Dublin (Baile Átha Cliath)	Dublin Map	540,000	Ireland
Helsinki (Helsingfors)	Helsinki Map	500,000	Finland
Kiev (Kyïv)	Kiev Map	2,650,000	Ukraine
Lisbon (Lisboa)	Lisbon Map	670,000	Portugal
Ljubljana	Ljubljana Map	280,000	Slovenia
London	London Map	7,000,000	United Kingdom
Luxembourg (Luxemburg, Letzebuerg)	Luxembourg City Map	78,000	Luxembourg
Madrid	Madrid Map	3,000,000	Spain
Minsk	Minsk Map	1,700,000	Belarus
Monaco	Monaco Map	28,000	Monaco
Moscow (Moskva)	Moscow Map	8,500,000	Russia

Flag	Name	Capital	Currency
	Netherlands	Amsterdam	Euro US dollar NA guilder Aruban florin
	Andorra	Andorra la Vella	Euro
	Greece	Athens	Euro
	Serbia	Belgrade	Serbian dinar
	Germany	Berlin	Euro
	Switzerland	Bern	Swiss franc
	Slovakia	Bratislava	Euro
	Belgium	Brussels	Euro

Figure 3.1: Examples of relational HTML tables on the Web. Sources: 1) `http://www.nationsonline.org` (left) , 2) `http://en.wikipedia.org` (right).

many concepts, or which concepts, are expected to be found in this data? While answering these questions we developed an open-domain, unsupervised information extraction technique named WebSets for extracting concept-instance pairs from a HTML table corpus.

3.1.1 Proposed Method

The techniques proposed in this section rely solely on HTML tables to detect coordinate terms[2]. The hypothesis here is that entities appearing in a table column possibly belong to the same concept, and if a set of entities co-occur together in many table columns coming from multiple URL domains, there is a high probability that they represent a coherent entity set. Given this hypothesis, each cell in a table becomes a potential entity and each table column becomes a potential entity set.

We proposed a representation based on co-occurrence of entity-triplets in table columns. It turned out that this representation helped us disambiguate multiple senses of the same entity string e.g., a triplet "Apple, Banana, Grapes" had different co-occurrence patterns when compared to "Apple, Google, Microsoft".

Next, we developed an efficient bottom-up clustering algorithm (Algorithm 1) which goes through the entire dataset only once and produces extremely precise (cluster purity 83-99%) coordinate-term clusters. The clusterer scans through each triplet record t which has occurred in at least *minUniqueDomain* distinct domains. A triplet and a cluster are represented with the same data-structure: (1) a set of entities, (2) a set of columnIds in which the entities co-occurred and (3) a set of domains in which the entities occurred.

The clusterer compares the overlap of triplet t against each cluster C_i. The triplet t is added

[2]Entities belonging to the same type are called coordinate terms. For example, "Pittsburgh" and "Seattle" are coordinate terms as both of them are of type "City".

to the first C_i so that either of the following two cases is true:
(1) at least 2 entities from t appear in cluster C_i i.e., $minEntityOverlap = 2$, and
(2) at least 2 columnIds from t appear in cluster C_i i.e., $minColumnOverlap = 2$.
In both these cases, intuitively there is a high probability that t belongs to the same category as cluster C_i. If no such overlap is found with existing clusters, the algorithm creates a new cluster and initializes it with the triplet t.

This clustering algorithm is order dependent, i.e., if the order in which records are processed changes, it might return a different set of clusters. Finding the optimal ordering of triplets is a hard problem, but a reasonably good ordering can be easily generated by ordering the triplets in the descending order of number of distinct domains. We discard triplets that appear in less than *minUniqueDomain* domains.

This clustering method outperforms K-means [126] in terms of Purity, and FM index (metrics are defined in Section 3.1.2). Further, the time complexity of our clustering algorithm is only $O(N * logN)$ (N being total number of HTML table cells in the corpus), making it more efficient and scalable than K-means[3] or agglomerative clustering[4]. We also presented a new method for combining candidate concept-instance pairs and coordinate-term clusters, and used it to suggest hypernyms for entity clusters (Algorithm 2).

Figure 3.2 summarizes the architecture of proposed WebSets system, which extracts concept-instance pairs from HTML tables in a given corpus. WebSets is composed of four main components: the *Table Parser*, the *Triplet Store Builder*, the *Bottom-Up Clusterer*, the *Hypernym Recommender*. Given a set of HTML pages, the *Table Parser* extracts potentially useful tables using a set of simple hand-coded heuristics. The *Triplet Store Builder* then goes over every column of every extracted table, and builds a dataset of entity triplets found in them. Each record of triplet store contains entities in the triplet, all (TableID, ColumnId) in which they co-occurred, and all domains in which they co-occurred. These triplets are then ranked by the number of distinct domains in which they co-occurred. The *Bottom-Up Clusterer* then clusters these triplets into consistent sets. This component considers only those triplets that have occurred in at least k domains, where k is a small constant. The *Hypernym Recommender* then considers each cluster, and recommends candidate hypernyms based on the overlap between the cluster and hypernym pattern dataset. More details about these methods and experiments is presented in Appendix A.

[3]K-Means takes $O(n * k * d * i)$ time where n is the number of d dimensional datapoints, k is number of clusters and i is the number of iterations to converge. Note that our proposed WebSets algorithm is single pass (not iterative).

[4]In the general case, the complexity of agglomerative clustering is $O(n^3)$. However, for some special cases, optimal efficient agglomerative methods have complexity $O(n^2)$.

Algorithm 1 Bottom-Up Clustering Algorithm.

function Bottom-Up-Clusterer ($TripletStore$) :$Clusters$ {Triplet records are ordered in descending order of number of distinct domains.}
Initialize $Clusters = \phi$; $max = 0$
for (every $t \in TripletStore$: such that
    $|t.domains|$ >=*minUniqueDomain*) **do**
  assigned = false
  for every $C_i \in Clusters$ **do**
    if $|t.entities \cap C_i.entities| >= minEntityOverlap$ OR $|t.col \cap C_i.col| >= minColumnOverlap$ **then**
      $C_i = C_i \cup t$
      $assigned = true$
      break;
    end if
  end for
  if not assigned **then**
    increment max
    Create new cluster $C_{max} = t$
    $Clusters = Clusters \cup C_{max}$
  end if
end for
end function

Algorithm 2 Hypernym Recommendation Algorithm.

function GenerateHypernyms
Given: c: An entity cluster generated by Algorithm 1,
I: Set of all entities ,
L: Set of all labels,
$H \subseteq L \times I$: Hyponym-concept dataset,
Returns: RL_c : Ranked list of hypernyms for c.
Algorithm:
$RL_c = \phi$
for every label $l \in L$ **do**
  H_l = Set of entities which co-occurred with l in H
  $Score(l) = |H_l \cap c|$
  $RL_c = RL_c \cup < l, Score(l) >$
end for
Sort RL_c in descending order of $Score(l)$
Output RL_c
end function

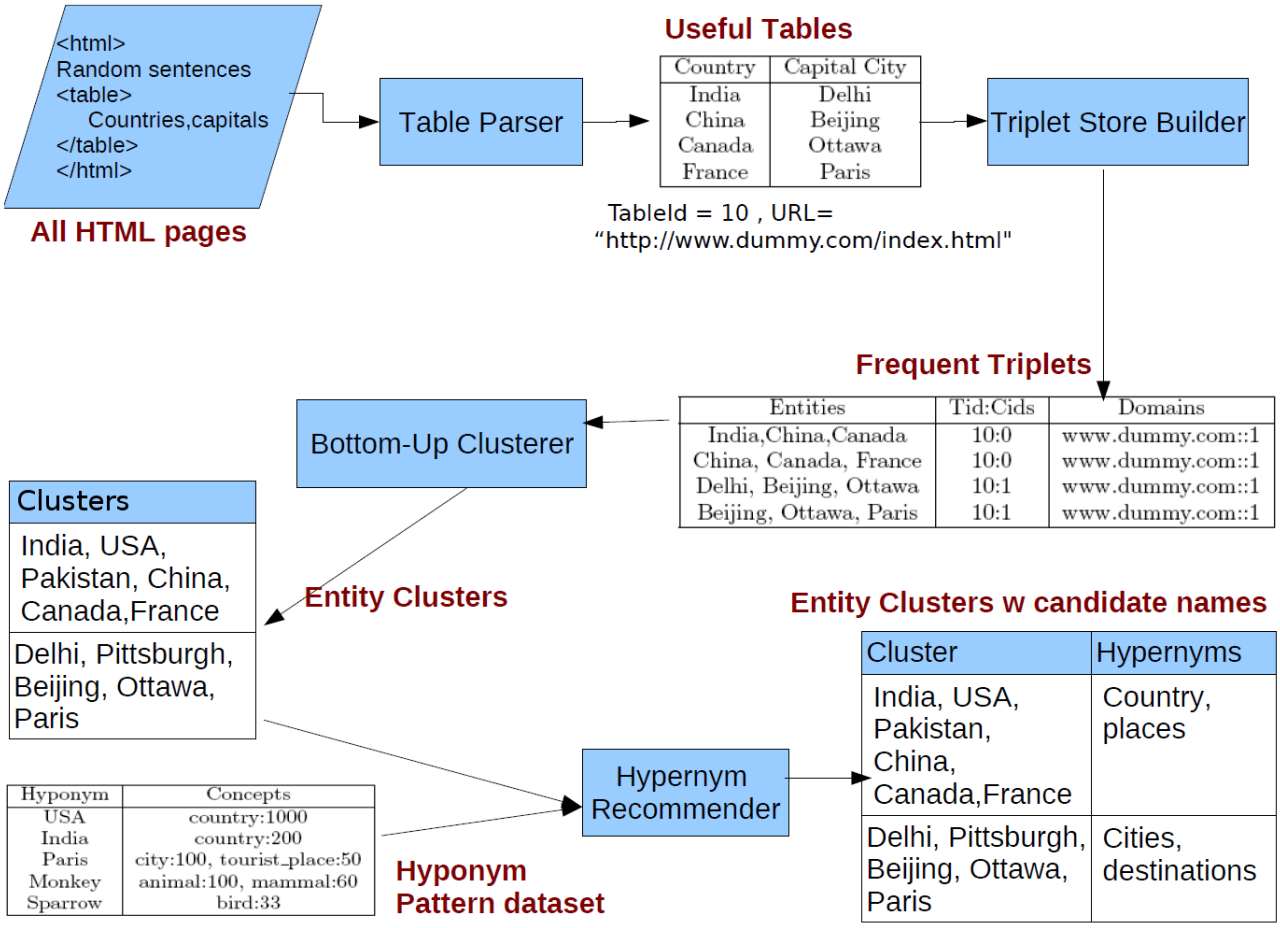


Figure 3.2: Architecture of the WebSets system.

3.1.2 Experimental Results

Here we describe the experimental results of applying WebSets techniques on various HTML table datasets extracted from the Web.

Datasets

To test the performance of WebSets, we created several webpage datasets that are likely to have coherent sets of entities. An evaluation can then be done to check whether the system extracts the expected entity sets from those datasets.

The first two datasets are created by crawling webpages that are tagged with particular topics in the delicious.com website [139]. The next four datasets are created using the SEAL [25, 137] and ASIA [136] systems which extract information from semi-structured pages on the Web. Each of these systems takes as input a name or seed examples of a category, and finds possible instances of that category using set expansion techniques. The last dataset is sampled from the ClueWeb09 corpus. We expect these datasets to be rich in semi-structured data.

1. **Delicious_Sports:** This dataset is a subset of the DAI-Labor Delicious corpus [139] which contains all public bookmarks of about 950,000 users retrieved from 'delicious.com' between December 2007 and April 2008. Delicious_Sports is created by taking only those URLs which are tagged as "sports".
2. **Delicious_Music:** This dataset is a subset of the DAI-Labor Delicious corpus [139], created by taking only those URLs which are tagged as "music".
3. **CSEAL_Useful:** CSEAL is one of the methods which suggests new category and relation instances to NELL. It mostly extracts entities out of semi-structured information on the Web. CSEAL_Useful dataset is a collection of those HTML pages from which CSEAL gathered information about entities in the NELL KB.
4. **Toy_Apple:** This is a small toy dataset created with the help of multiple SEAL queries such that it contains instances of word "Apple" as a fruit and "Apple" as a company. Through this dataset we tackle the challenge of separating multiple senses of a word. We also studied how different clustering algorithms and different entity representations perform on the clustering task.
5. **ASIA_NELL:** ASIA [136] extracts instances of a given semantic class name (e.g., 'car makers' to 'Ford, Nissan, Toyota ...') in an unsupervised fashion. It extracts set instances by utilizing Hearst patterns [61] along with the state-of-the-art set expansion technique implemented in SEAL. ASIA_NELL dataset is collected using hypernyms associated with entities in the NELL KB as queries for ASIA. Examples of such hypernyms are "City", "Bird", "Sports team" etc.
6. **ASIA_INT:** This dataset is also collected using the ASIA system but with another set of category names as input. These category names come from the intelligence domain. Examples of categories in this domain are "government types", "international organizations", "federal agencies", "religions" etc.
7. **Clueweb_HPR:** This dataset is collected by randomly sampling high quality pages in the

Dataset	#HTML pages	#tables
Toy_Apple	574	2.6K
Delicious_Sports	21K	146.3K
Delicious_Music	183K	643.3K
CSEAL_Useful	30K	322.8K
ASIA_NELL	112K	676.9K
ASIA_INT	121K	621.3K
Clueweb_HPR	100K	586.9K

Table 3.1: Dataset Statistics.

ClueWeb09 dataset [23]. For this purpose we used the Fusion spam scores [68] provided by Waterloo university and used pages with spam-rank score higher than 60% (higher score corresponds to lesser spam). We sampled uniformly at random 100K webpages from this set.

Table 3.1 shows the number of HTML pages and tables present in each of the above mentioned datasets. All these datasets are made publicly available for other researchers [126].

Relational Table Identification

Here we describe the data preprocessing step that involves table extraction from a HTML corpus, constituting the first stage of our WebSets system (described in Chapter 3). From the datasets described in Section 3.1.2, we parsed tables defined by HTML <table> tags[5] This is only a fraction of structured data available on the Web. Use of advanced table extraction techniques like Gatterbauer et al. [51] can provide more input data for our information extraction algorithms.

Further only a small fraction of HTML tables actually contain useful relational data. The remaining tables are used for formatting or rendering purposes rather than to present relational data. To filter out useful tables, we used the following set of features: (1) the number of rows (2) the number of non-link columns (3) the length of cells after removing formatting tags (4) whether table contains other HTML tables. Statistics of table identification for each dataset along with thresholds used for each of the above mentioned features are described in Appendix A.2.1.

Evaluation Metrics

Here we describe existing evaluation metrics used for clustering tasks. We compare the clustering algorithms in terms of commonly used clustering metrics: cluster purity (Purity), normalized mutual information (NMI), rand index (RI) [82] and Fowlkes-Mallows index (FM) which are defined as follows:

- **Cluster Purity (Accuracy):** To compute cluster purity, for each cluster, the class which is most frequent in it gets assigned. The accuracy of this assignment is then measured by counting the number of correctly assigned documents and dividing by the total number of documents. To compute cluster purity, the dataset must be fully labeled. Purity is defined

[5]We found that large number of HTML pages have broken syntax. We use the HTML syntax cleaning software Tidy [1] to fix the syntax in these pages.

as

$$purity(\Omega, C) = \frac{1}{N} \sum_k \max_j |\omega_k \bigcap c_j|$$

where N is total number of documents, $\Omega = \omega_1, \omega_2, ..., \omega_K$ is the set of clusters and $C = c_1, c_2, ..., c_J$ is the set of true classes. We interpret ω_k as the set of documents in cluster ω_k and c_j as the set of documents in cluster c_j.

- **Normalized Mutual Information:** High purity is easy to achieve when the number of clusters is large, hence using purity it is difficult to evaluate the trade offs in the quality of the clustering against the number of clusters. NMI [117] is a measure that allows us to evaluate such trade-off. Normalized mutual information can be computed on a partially labeled dataset as follows:

$$NMI(\Omega, C) = \frac{I(\Omega; C)}{[H(\Omega) + H(C)]/2}$$

where maximum likelihood estimates for mutual information and entropy are computed as follows:

$$I(\Omega; C) = \sum_k \sum_j \frac{|\omega_k \bigcap c_j|}{N} * log \frac{N * |\omega_k \bigcap c_j|}{|\omega_k| * |c_j|}$$

$$H(\Omega) = -\sum_k \frac{|\omega_k|}{N} * log \frac{|\omega_k|}{N}$$

- **Rand Index:** This metric [107] is based on an information-theoretic interpretation of clustering. Clustering is a series of decisions, one for each of the $N(N-1)/2$ pairs of documents in the collection. To compute this measure, a ground truth clustering for all datapoints is needed. True label denotes the pair belongs to same class or not. Predicted label denotes whether the pair belongs to same cluster or not. This way we can count True Positive (TP), False positive (FP), True Negative (TN) and False Negative (FN) score for the clustering. Rand Index is then defined as follows:

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

Thus RI is the accuracy of the decision to put each pair of data points in the same or different clusters.

- **Fowlkes-Mallows index (FM):** While the previous three metrics are applicable only for hard clusterings, the FM metric is defined for scenarios where a clustering approach produces overlapping clusters but the true classes are non-overlapping. Note that we are partitioning entity triplets, hence the clusters can be overlapping in terms of entities present in them. Let n_{ij} be the size of intersection of cluster ω_i and class c_j, $n_{i*} = \sum_j n_{ij}$ and $n_{*j} = \sum_i n_{ij}$. Then the Fowlkes-Mallows index is defined as:

$$FM = \frac{\sum_{ij} \binom{n_{ij}}{2}}{\sqrt{\sum_i \binom{n_{i*}}{2} \sum_j \binom{n_{*j}}{2}}}$$

The derivation of this formula can be found in Ramirez et al. [106].

Experimental Methodology

Note that evaluating unsupervised techniques is challenging, since it is very expensive to label each table-column of every dataset. Here we present a sampling based evaluation method which we used to evaluate WebSets on four datasets: CSEAL_Useful, ASIA_NELL, ASIA_INT and Clueweb_HPR. We chose these four datasets to cover the different types, i.e., domain-specific, open-domain and completely heterogeneous datasets. There are two parts of the evaluation: evaluating the labels of each cluster and checking whether each entity of that cluster is coherent with the assigned label.

The more subjective evaluation is of the form "deciding whether a cluster is meaningful or noisy", "assigning label to an unlabeled cluster". This was done by us (referred to as evaluators). The objective evaluation of the kind "whether X belongs to category Y" was done using Amazon Mechanical Turk. We created yes/no questions of the form "Is X of type Y?".

To evaluate the precision of clusters created by WebSets, we uniformly sampled a maximum of 100 clusters per dataset, with a maximum of 100 samples per cluster, and gave them to the Mechanical Turk in the form of yes/no questions. Each question was answered by three different individuals. The majority vote for each question was considered as a decision for that question. To evaluate the quality of Mechanical Turk labels, we sampled 100 questions at random, and manually answered the questions. Then we checked whether majority vote by the Mechanical Turk matched our answers. We specifically checked majority votes for some confusing questions which were likely to get labeled wrong. We found that the majority vote of three individuals was correct more than 90% of the time and in case of ambiguous questions precision estimates are biased low.

Results

We evaluate WebSets using three criteria: (1) Are the clusters generated by WebSets meaningful? (2) How good are the recommended hypernyms? (3) What is precision of the meaningful clusters? Subsequent sections discuss these experiments in detail.

Meaningfulness of Clusters: In this experiment, we did manual evaluation of meaningfulness of clusters. We uniformly sampled maximum 100 clusters from each dataset. We showed the following details of each cluster to the evaluator: (1) the top 5 hypernyms per cluster (2) a maximum of 100 entities sampled uniformly from the cluster.

We looked at the entities and checked whether any of the hypernyms suggested by the system is correct. If any one of them is correct then we labeled the cluster with that hypernym. If none of the hypernyms is correct, we labeled the cluster with any other hypernym of our choice that represents the cluster. If the entities in a cluster do not form any meaningful set, then the cluster is marked as "noisy". During the evaluation if any of the candidate hypernyms is picked as label or if a completely different label is chosen, then the cluster is considered as meaningful, else it is considered as noisy.

	#Triplets	#Clusters	#clusters with Hypernyms	% meaningful
CSEAL_Useful	165.2K	1090	312	69.0%
ASIA_NELL	11.4K	448	266	73.0%
ASIA_INT	15.1K	395	218	63.0%
Clueweb_HPR	561.0	47	34	70.5%

Table 3.2: Meaningfulness of generated clusters.

	#Clusters Evaluated	#Meaningful Clusters	#Hypernyms correct	MRR (meaningful)
CSEAL_Useful	100	69	57	0.56
ASIA_NELL	100	73	66	0.59
ASIA_INT	100	63	50	0.58
Clueweb_HPR	34	24	20	0.56

Table 3.3: Evaluation of Hypernym Recommender.

Table 3.2 shows that 63-73% of the clusters were labeled as meaningful. Note that number of triplets used by the clustering algorithm (Table 3.2) is different from total number of triplets in the triplet store (Table A.6), because only those triplets that occur in at least *minUniqueDomain* (set to 2 for all experiments) distinct domains are clustered.

Performance of Hypernym Recommendation: In this experiment, we evaluate the performance of the hypernym recommendation using the following criterion:
(1) What fraction of the total clusters were assigned some hypernym?: This can be directly computed by looking at the outputs generated by hypernym recommendation.
(2) For what fraction of clusters did the evaluator chose the label from the recommended hypernyms?: This can be computed by checking whether each of the manually assigned labels was one of the recommended labels.
(3) What is Mean Reciprocal Rank (MRR) of the hypernym ranking?: The evaluator gets to see ranked list of top 5 labels suggested by the hypernym recommender. We compute MRR based on rank of the label selected by the evaluator. While calculating MRR, we consider all meaningful clusters (including the ones for which label does not come from the recommended hypernyms).

Table 3.3 shows the results of this evaluation. Out of the random sample of clusters evaluated, hypernym recommendation could label 50-60% of them correctly. The MRR of labels is 0.56-0.59 for all the datasets.

Next we compare our hypernym recommendation technique with Van Durme and Pasca technique [132]. Our method is different from Van Durme and Pasca Method (DPM) in the sense that, they output a concept-instance pair only when it appears in a set of candidate concept-instance pairs, i.e., it exists in Hyponym-concept dataset. In our method, based on overlap of coordinate term cluster with the Hyponym Concept dataset, we extend the labels to the whole cluster. Another difference is the coordinate term clusters we are dealing with. DPM assumes term clusters are semantic partitions of terms present in the text corpus. The clusters generated by WebSets are clusters of table columns. There is higher possibility of having multiple small size clusters which all belong to same semantic class, but were not merged due to our single pass bottom-up clusterer. Hence the same method may not work equally well on these clusters.

Method	K	J	%Accuracy	yield (#pairs produced)	#correct pairs (predicted)
DPM	inf	0.0	34.6	88.6K	30.7K
DPM	5	0.2	50.0	0.8K	0.4K
DPMExt	inf	0	21.9	100,828.0K	22,081.3K
DPMExt	5	0.2	44.0	2.8K	1.2K
WebSets	-	-	67.7	73.7K	45.8K
WebSets-Ext	-	-	78.8	64.8K	51.1K

Table 3.4: Comparison of various methods in terms of accuracy and yield on CSEAL_Useful dataset.

Dataset	#Meaningful clusters evaluated	% Precision
CSEAL_Useful	69	98.6%
ASIA_NELL	73	98.5%
ASIA_INT	63	97.4%
Clueweb_HPR	24	99.0%

Table 3.5: Average precision of meaningful clusters.

We sample 100 concept-instance pairs randomly from output of each method to measure accuracy. The results of different methods are presented in Table A.9. As we can see, DPM generates concept-instance pairs with 50% accuracy even when run with conservative thresholds like $K = 5$ and $J = 0.2$. We also tried an extension of DPM called DPMExt which outputs a label for each entity in the cluster, when the label satisfies thresholds defined by J and K. This extension increases coverage of concept-instance pairs from 0.4K to 1.2K at the cost of a slight decrease in accuracy (50% to 44%).

The hypernym recommendation of WebSets, described in Algorithm 2, and only the topmost hypernym in the ranked list is produced for each cluster. Table A.9 shows that WebSets has a reasonable accuracy (62.2%) and yield compared to DPM and DPMExt. The Hyponym-concept dataset might contain noisy pairs. We also tried an extension of WebSets called WebSets-Ext which overcomes this problem by considering only those class-instance pairs which have occurred at least 5 times in the corpus. Adding this simple constraint improves accuracy from 67% to 78% and correct pairs yield from 45K to 51K.

Precision of Meaningful Clusters: In this experiment we want to evaluate the coherence of the clusters; i.e., whether all entities in a cluster are coherent with the label assigned to the cluster. To verify this, we evaluated the meaningful clusters found in the previous experiment, using the Mechanical Turk. This evaluation procedure is already discussed in Section 3.1. Table 3.5 shows that meaningful clusters[6] have precision in the range 97-99%. This indicates that WebSets generates coherent entity clusters and hypernym assignment is reasonable.

Entity Record vs. Triplet Record Representation: Next we compare the performance of both the

[6]First three values in column 2 of Table 3.5 are same as percentage values in column 5 of Table 3.2. This is because we sample maximum 100 clusters per dataset for the evaluation, hence percentage of meaningful clusters equals the actual number. For the Clueweb_HPR dataset, there are only 47 clusters in total, so all are evaluated.

Method	K	FM w/ Entity records	FM w/ Triplet records
WebSets		0.11 (K=25)	0.85 (K=34)
K-Means	30	0.09	0.35
	25	0.08	0.38

Table 3.6: Comparison of performance on Entity vs. Triplet record representation (Toy_Apple dataset).

WebSets and K-Means clustering algorithms on entity record vs. triplet record representation. In this experiment we use the Toy_Apple dataset (since this dataset has ambiguous noun-phrases like Apple). The entity representation contains a table-occurrence feature vector for each entity, while the triplet representation has a table-co-occurrence feature vector for a set of 3 entities. Both algorithms do hard clustering of entities/triplets (depending on the representation used), and also yield a soft-clustering of the corresponding table columns. To make all four algorithms comparable, we measure their performance in terms of clustering of table columns. Each table-column can be present in multiple clusters, but belongs to only one gold standard class (manual labels). Purity, NMI and RI metrics are not applicable for soft clustering, however the FM metric is valid. We ran K-Means algorithm for different values of K. Table A.8 shows the best performing results of K-Means. WebSets produced 25 clusters on entity record representation and 34 clusters using triplet representation. In terms of FM index, each method gives better performance on triplet record representation when compared to entity record representation. Hence the triplet record representation does improve these clustering methods.

3.1.3 Application: Summary of the Corpus

The sets of entities produced by WebSets can be considered as a summary of the document corpus in terms of what concepts and entities are discussed in the corpus. Tables 3.7 and 3.8 show such summaries for the datasets ASIA_INT and Delicious_Music respectively. We choose these datasets because they are domain-specific and hence we have some idea of what entity sets are expected to be found. Looking at the summary we can verify if those expectations are met. Due to space constraints only few clusters with ten entities from each of them are presented here. The clusters are hand picked, however the cluster labels are taken as the top label predicted by our hypernym recommendation component (Algorithm 2).

3.2 Related Work: IE from Tables, Semi-supervised Learning

HTML tables on the Web have received lot of attention in past few years. Gatterbauer et al. [51] focus on extracting tabular data from various kinds of HTML pages and table-like visual representations. The WebTables [19] system extracted schema information from a huge corpus of 14.1 billion HTML tables from Google's general-purpose web crawl. They built an attribute correlation statistics database (AcsDB), which can be used to create an attribute name thesaurus and a schema auto-completion system. Gupta et al. [56] focus on the task of extending a table given a few seed rows. Their technique consolidates HTML lists relevant to the example rows

Religions: Buddhism, Christianity, Islam, Sikhism, Taoism, Zoroastrianism, Jainism, Bahai, Judaism, Hinduism, Confucianism
Government: Monarchy, Limited Democracy, Islamic Republic, Parliamentary Self Governing Territory, Parliamentary Republic, Constitutional Republic, Republic Presidential Multi-party System, Constitutional Democracy, Democratic Republic, Parliamentary Democracy
International Organizations: United Nations Children Fund UNICEF, Southeast European Cooperative Initiative SECI, World Trade Organization WTO, Indian Ocean Commission INOC, Economic and Social Council ECOSOC, Caribbean Community and Common Market CARICOM, Western European Union WEU, Black Sea Economic Cooperation Zone BSEC, Nuclear Energy Agency NEA, World Confederation of Labor WCL
Languages: Hebrew, Portuguese, Danish, Anzanian, Croatian, Phoenician, Brazilian, Surinamese, Burkinabe, Barbadian, Cuban

Table 3.7: Example Clusters from ASIA_INT.

Instruments: Flute, Tuba , String Orchestra, Chimes, Harmonium, Bassoon, Woodwinds, Glockenspiel, French horn, Timpani, Piano
Intervals: Whole tone, Major sixth, Fifth, Perfect fifth, Seventh, Third, Diminished fifth, Whole step, Fourth, Minor seventh, Major third, Minor third
Genres: Smooth jazz, Gothic, Metal rock, Rock, Pop, Hip hop, Rock n roll, Country, Folk, Punk rock
Audio Equipments: Audio editor , General midi synthesizer , Audio recorder , Multichannel digital audio workstation , Drum sequencer , Mixers , Music engraving system , Audio server , Mastering software , Soundfont sample player

Table 3.8: Example Clusters from Delicious_Music.

to build a result table. Gupta and Sarawagi [57] consider jointly training structured extraction models from overlapping web source (primarily in tables), thus avoiding the need for labeled data. Our approach, WebSets does not require seed examples, but instead extracts concept-instance pairs in an unsupervised manner from a corpus. Limaye et al. [79] proposed a system to use an existing catalog and type hierarchy for annotating table columns and cells. WebSets differs in that the Hearst-pattern data it uses is noisier than a catalog or type hierarchy.

Many systems perform semi-supervised information extraction from free text on the web, using only a few seed examples or seed rules. These systems include KnowItAll [41, 42], ASIA [136], and Coupled Pattern Learning (CPL) [25]. Along similar lines, Parameswaran et al. [101] propose a concept extraction algorithm which can identify a canonical form of a concept, filtering out sub-concepts or super-concepts; and Ritter et al. [110] describe a scheme for filtering concept-instance pairs, using a SVM classifier which uses as features frequency statistics for several Hearst patterns on a large corpus. Given a training set of text containing known concept-instance pairs, Snow et al. [115] learns "dependency path" features, which can further expand the set of concept-instance pairs. WebSets is different from most of these approaches in that it builds all sets of entities from a given corpus, and does not require seed sets, a starting ontology, or a set of target concept names.

TextRunner [145] is an open-domain IE system which makes a single pass over the corpus of unstructured text and extracts a large set of relational tuples, without requiring any human input. Unlike WebSets, however, it does not build coherent sets of category or relation instances. Pantel

and Ravichandran [100] proposes a method to automatically label distributional term clusters using Hearst-like patterns, and Van Durme and Pasca [132] proposed an alternative approach method to extract labeled classes of instances from unstructured text. These approaches use only unstructured text to find coordinate terms and assigning hypernyms, whereas WebSets uses HTML tables. As we show in the experimental results, WebSets uses a novel method for combining coordinate-term clusters and hypernyms data that quantitatively improves over Van Durme and Pasca's method for combining coordinate-term clusters and hypernym data.

There also exist some systems that use both free-text and tabular information. Talukdar et al. [121] proposed a graph random walk based semi-supervised label propagation technique for open domain class instance extractions, which extends the system of Van Durme and Pasca by using table data (from WebTables) as well as free-text distributional clusters. However, unlike Talukdar et al.'s system, WebSets does not require distributional clusters. Coupled SEAL (CSEAL) [25] use of mutual exclusion, containment and type checking relationships to extend SEAL, and CPL [25] and CSEAL are the two components of NELL [91], a multi-strategy semi-supervised learning system. However, unlike NELL, WebSets does not require seed instances or an ontology as input. Shinzato and Torisawa [113] showed that coordinate terms can be extracted from itemizations in structured web documents. Their method finds concept-instance pairs pertaining to a single query list, and finds candidate hypernyms by querying the web on-the-fly to collect documents. In contrast, WebSets processes all lists in a corpora simultaneously, and makes no web queries.

In the recent years there has been more work in the area of information extraction from semi-structured data. Balasubramanyan et al. [6] used the HTML tables data extracted by WebSets as a multi-view dataset to learn a semi-supervised mixed membership model that can make use of feature and document labels. Adelfio and Samet [3] developed a method to extract schema information from tables found on the webpages and in spreadsheets. They used conditional random fields to determine schema including row groupings. Instead of HTML table structure, some recent techniques use DOM trees to extract information from structured webpages. Insa et al. [63] used DOM trees to analyze the hierarchical relationship between the webpage element along with the distribution of textual information to do content extraction. Castillo et al. [27] worked on a novel online technique that can extract information from a set of interconnected webpages based on DOM distances.

3.3 Semi-supervised Bottom-Up Clustering to Extend the NELL Knowledge Base

The bottom-up clustering algorithm proposed in Section 3.1 assumes no human input. However, there is valuable information present in the existing knowledge bases about the concepts and example instances of those concepts that are expected to be present in the Web data. In this section we will discuss, how to use this seed data to guide a bottom-up clustering algorithm.

3.3.1 Semi-supervised WebSets

We changed the *Bottom-Up Clusterer* (Algorithm 1) slightly for the task of populating an existing knowledge base. The Clusterer initializes $Clusters$ with existing categories and their instances in the knowledge base. With this simple modification *Bottom-Up Clusterer* starts from NELL KB categories and ends up with clusters which contain additions made to already existing KB clusters and suggestions of some new clusters. For experiments in this section, we use around a dozen seed examples in each category of NELL for initialization. New entities added to existing NELL KB clusters are called "promotions". This version of the algorithm uses facts already learnt by a knowledge base, hence categorized as a semi-supervised approach.

3.3.2 Experimental Evaluation using NELL

The set of experiments explained in this section, will enable us to answer the following questions: (1) How precise are the recommendations made for existing categories of a knowledge base? (2) What is the coverage of WebSets system? Does it find the expected sets of entities? (3) How meaningful and precise are the new categories suggested for a knowledge base?

To answer these questions, we ran the WebSets system in semi-supervised mode. Coverage is measured by looking at promotions made by WebSets and CSEAL for the same set of NELL categories. The manual evaluation process for measuring precision and meaningfulness of clusters is the same as discussed in Section 3.1.

Both CSEAL [25] and WebSets use structured information to come up with entity sets, but the processes they follow have some fundamental differences. CSEAL starts with seeds for each category and queries the Web for pages containing co-occurrence of seed instances. It then extracts more instances of same category from those pages by building wrappers, and the process continues. On the other hand, WebSets starts with a set of available HTML pages and comes up with sets that the system strongly believes are meaningful. These differences in approaches and nature of inputs indicate that direct comparison between WebSets and CSEAL is not possible, hence we compare them in terms of their usefulness in enhancing existing knowledge base.

Evaluation of existing NELL categories:

In this experiment, we compare the coverage of sets produced by WebSets and CSEAL. We did this evaluation for only those NELL categories for which results of CSEAL are available in Carlson et. al. [25] and WebSets found some promotions using the CSEAL_Useful dataset. The number of promotions made by both methods are shown in Table 3.9. To compute the precision

Category	#Promotions		%Precision	
	WebSets	CSEAL	WebSets	CSEAL
athlete	2	276	100	100
city	3117	368	84.5	97
coach	11	619	100	100
company	336	245	97	100
country	1569	130	60.2	97
hobby	7	77	71.4	77
scientist	423	928	88	100
sports team	88	864	97.7	87
state / province	106	114	53.9	83
Aggregate	5659	3621	83.6	93.4

Table 3.9: Number and precision of promotions by WebSets (using CSEAL_Useful dataset) and CSEAL (using the Web) to existing NELL categories.

of WebSets, we labeled 100 random samples from each cluster with the help of Mechanical Turk. Table 3.9 also shows the precision evaluation for both methods. From these results we can say that WebSets produces results with 83% precision on average lesser when compared to CSEAL which produces 93% precision.

In Table 3.10 we present the precision of promotions made across all four WebSets datasets to the NELL categories. As can be seen, for most of the clusters, precision is very high. WebSets gives noisy results for two clusters including "color" and "female names" when run on ASIA_INT. We did some error analysis to understand why the words like "they", "said", "no", "people", "are", get added to category "color". We found that several educational websites for children contain tables with a vocabulary of common words arranged alphabetically in HTML tables, which violates our assumption that entities occurring in a table column belong to a coherent concept.

Evaluation of new category suggestions:
In this section, we evaluate both meaningfulness and precision of the additional clusters proposed by WebSets. These are the clusters which could not get merged with NELL clusters due to lack of entity overlap or absence of the category in NELL KB. To judge meaningfulness of these clusters we manually labeled each cluster by looking at the entities in it. If the cluster does not represent any consistent set, it is marked as "noisy".

Table 3.11 shows these evaluation results for datasets CSEAL_Useful and ASIA_INT. We can see that 81 out of 107 clusters from CSEAL_Useful are meaningful with average precision within those clusters being 94.44%. However for ASIA_INT, a relatively smaller fraction of clusters (50 out of 126) are meaningful, but within meaningful clusters precision is quite high, i.e., 92.41%. Table 3.12 shows a sample of meaningful clusters from CSEAL_Useful and ASIA_INT. We can see that most of the clusters have very high accuracy (above 95%). These extra clusters can be added to the knowledge base to improve its category coverage.

The set of experiments explained in this section, will enable us to answer following questions: (1) How precise are the recommendations made for existing categories of a knowledge base? (2)

Category	Precision (%)			
	CSEAL_Useful	ASIA_NELL	ASIA_INT	Clueweb_HPR
automobile maker	-	100.00	83.33	-
automobile model	-	-	100.00	-
bird	80.00	100.00	-	-
blog	98.80	-	-	-
building material	-	91.30	84.62	-
cardgame	-	100.00	-	-
chemical	88.65	94.87	-	-
color	-	71.43	**29.35**	-
consumer electronic item	-	73.68	-	-
continent	82.35	100.00	76.92	-
day of week	-	-	-	100.00
ethnic group	97.78	-	-	-
female names	-	-	**20.00**	-
fish	94.85	94.44	-	-
insect	100.00	-	-	-
language	88.29	100.00	92.93	100.00
magazine	-	-	100.00	-
mountain	100.00	-	-	-
music genre	100.00	-	-	-
music instrument	-	100.00	-	-
newspaper	93.41	-	-	-
park	98.90	-	-	-
planet	100.00	-	-	-
politics blog	88.00	-	-	-
programming language	-	-	81.32	-
record label	55.56	50.00	-	-
religion	-	-	100.00	-
stadium / event venue	96.70	-	-	-
television network	97.80	-	-	-
television station	87.78	-	-	-
vegetable	-	100.00	-	-
video game	72.73	-	-	-
Average precision	90.61	90.44	76.85	100.00

Table 3.10: Performance of WebSets on NELL categories.

	#Clusters	#Meaningful	#Noisy	Avg. Precision (meaningful)
CSEAL_Useful	107	81	26	94.44
ASIA_INT	126	50	76	92.41

Table 3.11: Evaluation of meaningfulness of extra clusters suggested to NELL.

SEAL_Useful		ASIA_INT	
Manual label	Precision	Manual label	Precision
tv channels	100	airline	100
tourist place	100	apparel	100
shoes/footwear	100	car make	100
researcher	100	network protocol	100
months	100	government type	100
laptop	100	iraq event	100
language	100	occupation type	100
pond	100	police designation	100
celebrity	100	region	100
bird	100	religion	100
baseball players	100	contact information	100
athletics	100	university	100
actress/models	100	international organization	96.67
pond/lake	97.78	service	96.15
chemicals	97.78	department	93.33
product	96	linux jargon	92.86
mountain	91.67	province	88.24
genes	91.11	chemical element	87.38
book type	87.5	action	63.04
shoes	85.71	social metric	55.56
film studio	78.57	USA state	48.42
hobby	55.56	computer hardware	16.67

Table 3.12: Precision (%) of manually labeled extra clusters from CSEAL_Useful, ASIA_INT.

What is the coverage of WebSets system? Does it find the expected sets of entities? (3) How meaningful and precise are the new categories suggested for a knowledge base?

To answer these questions, we ran the WebSets system in semi-supervised mode. Coverage is measured by looking at promotions made by WebSets and other IE techniques (CPL and CSEAL) for same set of NELL categories. This method can enrich an existing knowledge base. Experiments show that our method improves the coverage of existing categories with 80-90% accuracy. It also suggests new categories that can be added to the knowledge base.

3.4 Lessons Learned from WebSets

Through the experiments on WebSets we learned multiple lessons. First, the triplet representation of entities is very useful. From the experimental results in Table A.8, we can see that the triplet representation helped both WebSets and K-Means clustering algorithms in terms of FMI metric. Further, the triplet representation could disambiguate "Apple" as fruit vs. "Apple" as company in the Toy_Apple dataset. This indicates that representing and clustering sets of three entities rather than an individual entity is beneficial, both in terms of disambiguating multiple senses and better soft clustering of table columns. However, it is easier to construct such data representation from table columns, or possibly lists, but it is not trivial to do so in a general text dataset. (Note that an entity triplet is not comparable to trigrams of words, as the implicit assumption in triplet representations is that all three entities potentially belong to a single class/concept, which is not necessarily true in case of word trigrams.)

However, this case study suggested three other potential research directions, described below.

Presence of Unanticipated Classes

Experiments with the WebSets system showed that the ability to dynamically add new clusters in a semi-supervised task is a powerful technique. During the manual labeling of clusters generated by semi-supervised WebSets (seeded with NELL categories), we observed that it created many extra clusters. From Table 3.12, there are newly discovered clusters like "police designation", "social metric", "government type" that are coherent concepts but not present in the NELL ontology. Thus WebSets can not only populate known concepts, but it can also discover coherent entity clusters belonging to potentially interesting concepts that can further be added to existing ontologies. It is evident that this technique is not restricted to any dataset/domain and hence we will explore it further in the general context of semi-supervised learning.

We did an experiment to check how robust semi-supervised clustering is w.r.t unknown clusters hidden in the data. For this purpose we worked on two datasets coming from different domains. The first dataset is Delicious_Sports which we extracted from HTML tables on the Web, and the second dataset is 20-Newsgroups which is a standard dataset for text categorization tasks. Both datasets are completely labeled, we then tried a semi-supervised K-Means algorithm on this dataset, by initializing the process with seeds of only a fraction of classes.

For example in Figure 3.3, for Delicious_Sports, we started the algorithm by providing seeds for only 5 out of 26 classes, and for 20-Newsgroups we provided seeds for 6 out of 20 classes.

Thus the algorithm knows many fewer classes than the actual number. We can see that semi-supervised K-Means is affected by number of unseen classes. As the number of EM iterations increases, it confuses more and more non-seed classes with the seeded classes. This phenomenon is more evident in the 20-Newsgroups dataset as the Delicious_Sports dataset is much smaller and cleaner. These confusion matrices illustrate potential semantic drift in such scenarios.

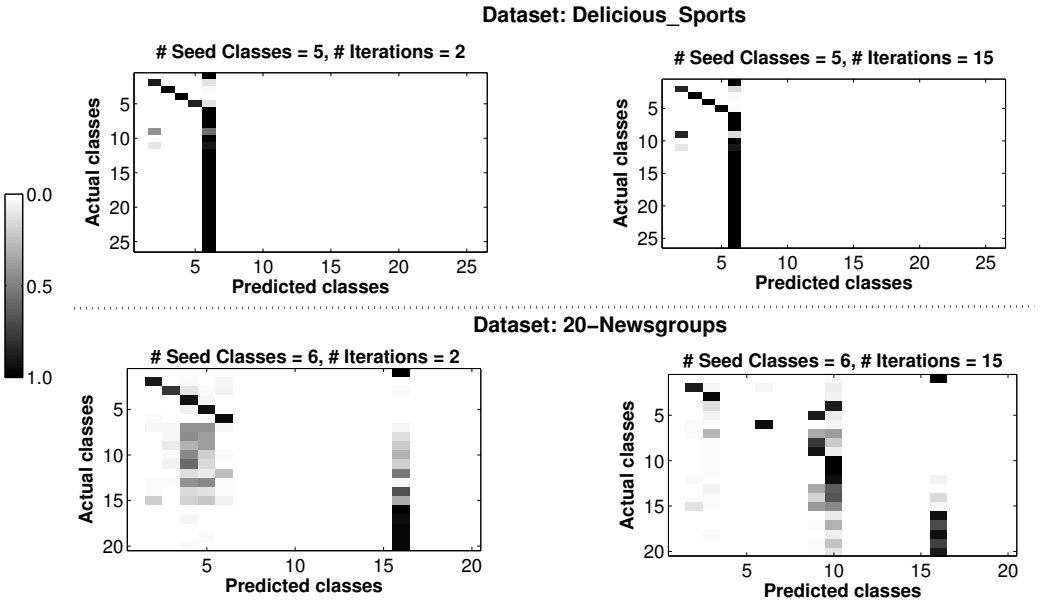


Figure 3.3: Confusion matrices varying number of EM iterations of the K-Means algorithm for Delicious_Sports and 20-Newsgroups datasets.

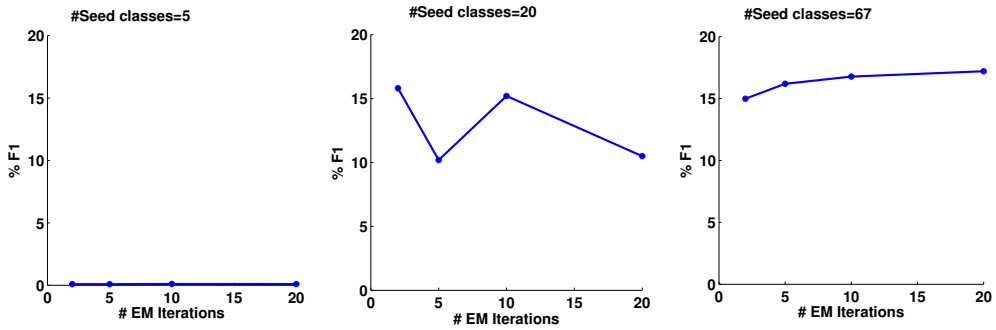
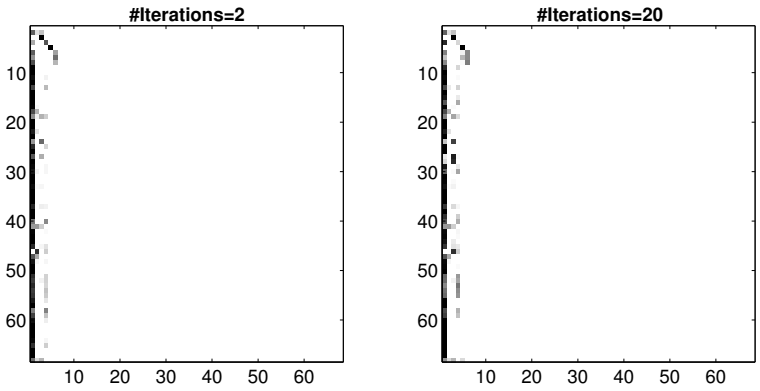
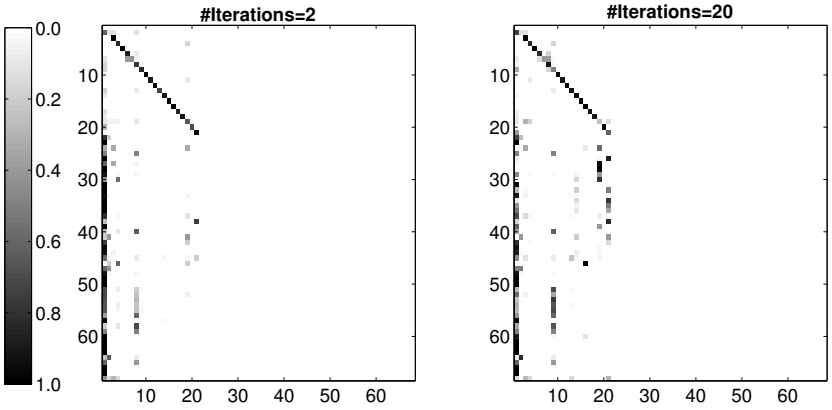


Figure 3.4: Macro Averaged F1 plots for semi-supervised K-Means on CSEAL_Useful dataset.

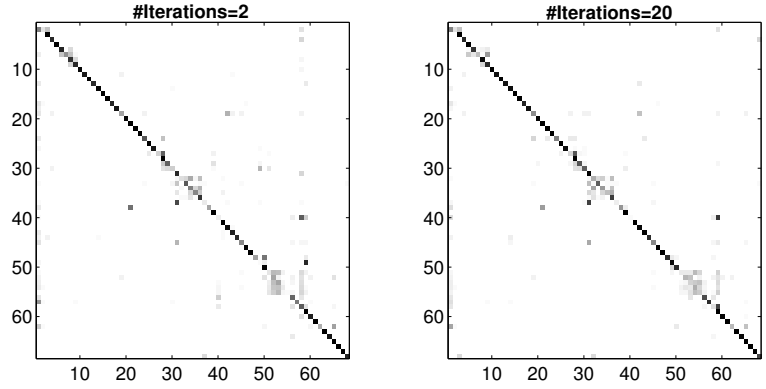
We did similar experiments with the CSEAL_Useful dataset, which contains datapoints belonging to 67 NELL categories. Figure 3.4 shows the performance of semi-supervised K-Means in terms of macro-averaged seed class F1 for varying number of seed classes. Figure 3.5 describes how the confusion matrices change with the number of EM iterations. We can see that when the number of seed classes are much smaller than actual, more EM iterations result in semantic drift for the seed classes. This problem and our proposed solution is described in depth in Chapter 4.



(a) #Seed classes = 5



(b) #Seed classes = 20



(c) #Seed classes = 67

Figure 3.5: Confusion matrices varying the number of EM iterations of the K-Means algorithm for the CSEAL_Useful dataset.

Multiple Data Views

For many information extraction tasks, including the knowledge base population for NELL, a noun phrase can be represented by features extracted from different kinds of data, e.g., occurrences of a noun phrases with text contexts, in HTML table columns, and morphological features can be considered as three different data views. In the WebSets system, we can construct multi-view features by representing each noun-phrase by 1) a set of co-occurring noun-phrases, 2) a set of table columns it appears in, 3) a set of hypernyms it appears with in the context of Hearst patterns, and 4) a set of domains it is extracted from. All these clues are useful and give us the information that can be leveraged to classify the noun-phrase more effectively.

In Chapter 5, we describe semi-supervised methods that can leverage multiple data views and solve a constrained optimization problem for each datapoint, to output labels that are consistent in terms of evidence from multiple views.

Leveraging Ontological Constraints

In the semi-supervised WebSets experiment, we only use the facts present in NELL KB as initial seeds. However, one can also use the ontology of classes and constraints between the classes to improve this kind of learning, e.g., the ontology can give us information of the kind: "each noun-phrase that is classified as Mammal should also be classified as Animal" (subset constraint) and "any noun-phrase that is classified as Mammal should not be classified as Reptile" (mutual exclusion constraint). In Chapter 6 we will explore techniques that can learn multi-class classifiers leveraging ontological constraints.

Chapter 4

Exploratory Learning: Semi-supervised Learning in the Presence of Unanticipated Classes

The semi-supervised bottom-up clustering algorithm discussed in the previous chapter can populate concepts from NELL as well as discover new clusters of entities that do not belong to any of the NELL classes. This revealed a novel problem relevant for such knowledge base population tasks, i.e., we know about some classes, and some instances of these classes, but the unlabeled datasets contain examples of many more classes that are not known upfront. Can an algorithm do both semi-supervised learning for the classes with training data and unsupervised clustering for classes without training data? To the best of our knowledge there is no existing technique that is targeted to solve this problem.

In this chapter, we propose an Exploratory Learning framework that can make use of limited supervision for some of the concepts to populate these known concepts as well as discover new ones. This work is published in the proceedings of ECML/PKDD 2013 [128].

4.1 Exploratory EM Algorithm

In multiclass semi-supervised learning (SSL), it is sometimes the case that the number of classes present in the data is not known, and hence no labeled examples are provided for some classes. In this section we present variants of well-known semi-supervised multiclass learning methods that are robust when the data contains an unknown number of classes. In particular, we propose an "exploratory" extension of expectation-maximization (EM) that explores different numbers of classes while learning.

Algorithm 3 presents a generic Exploratory EM algorithm (without specifying the model being used). There are two main differences between the algorithm and standard classification-EM approaches to SSL. First, in the E step, each of the unlabeled datapoints x is either assigned to one of the existing classes, or to a newly-created class. The intuition we use is that a new class should be introduced to hold x when the probability of x belonging to existing classes is close to uniform. This suggests that x is not a good fit to any of the existing classes, and that adding

x to any existing class will lower the total data likelihood substantially. Second, in the M-step of iteration t, we choose either the model proposed by Exploratory EM method which might have more classes than the previous iteration $t-1$, or the baseline version with same number of classes as iteration $t-1$. This choice is based on whether exploratory model satisfies a model selection criterion in terms of increased data likelihood and model complexity.

4.1.1 Discussion

Let us discuss the applicability of this algorithm for clustering as well as classification tasks. Notice that Algorithm 3 reverts to an unsupervised clustering method if X^l is empty, and reverts to a supervised generative learner if X^u is empty. Likewise, if no new classes are generated, then it behaves as a multiclass SSL method; for instance, if the classes are well-separated and X^l contains enough labels for every class to approximate these classes, then it is unlikely that the criterion of nearly-uniform class probabilities will be met, and the algorithm reverts to SSL. Henceforth we will use the terms "class" and "cluster" interchangeably.

Friedman [47] proposed the Structural EM algorithm that combines the standard EM algorithm, which optimizes parameters, with structure search for model selection. This algorithm learns networks based on penalized likelihood scores, in the presence of missing data. In each iteration it evaluates multiple models based on the expected scores of models with missing data, and selects the model with best expected score. This algorithm converges at local maxima for penalized log likelihood (the score includes penalty for increased model complexity). Similar to Structural EM, in each iteration of Algorithm 3, we evaluate two models, one with and one without adding extra classes. These two models are scored using a model selection criterion like AIC or BIC, and the model with best penalized data likelihood score is selected in each iteration. Hence we can say that, similar to Structural EM, the Exploratory EM algorithm will converge at local maxima for penalized log data likelihood.

Algorithm 3 EM algorithm for exploratory learning with model selection.

function Exploratory EM (X^l, Y^l, X^u, $\{C_1 \dots C_k\}$): $\{C_{k+1} \dots C_{k+m}\}$, θ_{k+m}, Y^u
Input: X^l labeled data points; Y^l labels of X^l; X^u unlabeled data points; $\{C_1 \dots C_k\}$ set of known classes to which x's belong.
Output: $\{C_{k+1} \dots C_m\}$ newly-discovered classes; $\{\theta_1, \dots, \theta_{k+m}\}$ parameters for k seed and m newly added classes; Y^u labels for X^u
{Initialize classifiers θ_j for class C_j using seeds provided for C_j}
$\theta_1^0 \dots \theta_k^0 = argmax_\theta L(X^l, Y^l)$
while class assignments AND #classes not converged **do**
  k_{old} is the #classes before E step.
  k_{cur} is the current #classes.
  Log-likelihood *BaseLL* $= logP(X|\theta^{(t)}_{k_{old}})$
  {E step: (Iteration t) Classify each datapoint at each level}
  for $i = 1$ to $|X^u|$ **do**
    Find $P(C_j|X_i, \theta_1^{(t)}, \dots, \theta^{(t)}_{k_{cur}})$ for all labels $1 \le j \le k_{cur}$
    if nearlyUniform($P(C_1|X_i), \dots, P(C_{k_{cur}}|X_i)$) **then**
      Increment k_{cur}; Let $C_{k_{cur}}$ be the new class.
      Label X_i with $C_{k_{cur}}$ in Y^u, and compute parameters $\theta^{(t)}_{k_{cur}}$ for the new class.
    else
      Assign X_i to $(argmax_{C_j} P(C_j|X_i))$ in Y^u where $1 \le j \le k_{cur}$
    end if
  end for
  $k_{new} = k_{cur}$ is the #classes after E step.
  Log-likelihood *ExploreLL* $= logP(X|\theta^{(t)}_{k_{new}})$
  {M step: Recompute model parameters based on $Y^{(t)}$ }
  if Model selection criterion (k_{old}, *BaseLL*, k_{new}, *ExploreLL*) selects exploratory model **then**
    {Adopt the new model with k_{new} classes}
    $\theta^{(t+1)}_{k_{new}} = argmax_\theta L(X^l, Y^l, X^u, Y^{u(t)}|\theta^{(t)}_{k_{new}})$
  else
    {Keep the old model with k_{old} classes}
    $\theta^{(t+1)}_{k_{old}} = argmax_\theta L(X^l, Y^l, X^u, Y^{u(t)}|\theta^{(t)}_{k_{old}})$
  end if
end while
end function

4.1.2 Model Selection

Any training error based model selection can be used in this framework e.g., BIC, AIC and AICc. Burnham and Anderson [18] have experimented with the AIC criteria and proposed AICc for datasets where the number of datapoints is less than 40 times number of features. The formulae for scoring a model using each of the three criteria that we tried are:

$$\begin{aligned} BIC(g) &= -2 * L(g) + v * ln(n) && (4.1)\\ AIC(g) &= -2 * L(g) + 2 * v && (4.2)\\ AICc(g) &= AIC(g) + 2 * v * (v+1)/(n-v-1) && (4.3) \end{aligned}$$

where g is the model being evaluated, $L(g)$ is the log-likelihood of the data given g, v is the number of free parameters of the model and n is the number of data-points, so when comparing two models, a lower value is preferred. The extended Akaike information criterion (AICc) suited best for our experiments since our datasets have large number of features and a smaller number of data points. With the AICc criterion, the objective function being optimized is:

$$\begin{aligned} &\max_{m,\{\theta^1 \ldots \theta^m\}, m \geq k} \{\text{Log Data Likelihood} - \text{Model penalty}\} \\ &\text{i.e.,} \max_{m,\{\theta^1 \ldots \theta^m\}, m \geq k} \{\log P(X|\theta^1, \ldots, \theta^m)\} - \{v + (v * (v+1)/(n-v-1))\} \end{aligned} \quad (4.4)$$

Here, k is the number of seed classes given as input to the algorithm and m is the number of classes in the resultant model ($m \geq k$).

4.1.3 Exploratory EM using Different Language Models

In this section we will consider variants of Exploratory EM algorithm with different language models. The choice of language model decides the method used in Algorithm 3 Line 10, 21 and 23.

Semi-Supervised Naive Bayes: Nigam et al. [98] proposed an EM-based semi-supervised version of multinomial Naive Bayes. In this model $P(C_j|x) \propto P(x|C_j) * P(C_j)$, for each unlabeled point x. The probability $P(x|C_j)$ is estimated by treating each feature in x as an independent draw from a class-specific multinomial. In document classification, the features are word occurrences, and the number of outcomes of the multinomial is the vocabulary size. This method can be naturally used as an instance of Exploratory EM, using the multinomial model to compute $P(C_j|x)$ in Algorithm 3 Line 10. The M step is also trivial, requiring only estimates of $P(w|C_j)$ for each word/feature w.

Seeded K-Means: It has often been observed that K-Means and EM are algorithmically similar. Basu and Mooney [9] proposed a seeded version of K-Means, which is very analogous to Nigam et al's semi-supervised Naive Bayes, as another technique for semi-supervised learning. Seeded K-Means takes as input a number of clusters, and seed examples for each cluster. The seeds are used to define an initial set of cluster centroids, and then the algorithm iterates between an

Algorithm 4 JS criterion for new class creation.

function JSCriterion($[P(C_1|x) \ \ldots \ P(C_k|x)]$):
Input: $[P(C_1|x) \ldots P(C_k|x)]$ probability distribution of existing classes for a data point x
Output: *decision* : **true** iff new class needs to be created
u = $[1/k \ldots 1/k]$ {i.e., the uniform distribution with current number of classes = k}
decision = false
if Jensen-Shannon-Divergence(u, $P(C_j|x)$) $< \frac{1}{k}$ **then**
    decision = true
end if
end function

"E step" (assigning unlabeled points to the closest centroid) and an "M step" (recomputing the centroids).

In the seeded K-Means instance of Exploratory EM, we again define $P(C_j|x) \propto P(x|C_j) * P(C_j)$, but define $P(x|C_j) = x \cdot C_j$, i.e., the inner product of a vector representing x and a vector representing the centroid of cluster j. Specifically, x and C_j both are represented as L_1 normalized[7] TFIDF feature vectors. The centroid of a new cluster is initialized with smoothed counts from x. In the "M step", we recompute the centroids of clusters in the usual way.

Seeded Von-Mises Fisher: The connection between K-Means and EM is explicated by Banerjee et al. [7], who described an EM algorithm that is directly inspired by K-Means and TFIDF-based representations. In particular, they describe generative cluster models based on the von Mises-Fisher (vMF) distribution, which describes data distributed on the unit hypersphere. Here we consider the "hard-EM" algorithm proposed by Banerjee et al, and use it in the seeded (semi-supervised) setting proposed by Basu et al. [9]. This natural extension of Banerjee et al[7]'s work can be extended to our exploratory setting.

As in seeded K-Means, the parameters of the vMF distribution are initialized using the seed examples for each known cluster. In each iteration, we compute the probability of C_j given data point x, using the vMF distribution, and then assign x to the cluster for which this probability is maximized. The parameters of the vMF distribution for each cluster are then recomputed in the M step. For this method, we use a TFIDF-based L_2 normalized vectors[8], which lie on the unit hypersphere.

Seeded vMF and seeded K-Means are closely related—in particular, seeded vMF can be viewed as a more probabilistically principled version of seeded K-Means. Both methods allow use of TFIDF-based weighted unigram representations, which are often preferable to TF weighted unigram representations for text [76].

4.1.4 Strategies for Inducing New Clusters/Classes

In this section we will formally describe some possible strategies for introducing new classes in the E step of the algorithm. They are presented in detail in Algorithms 4 and 5, and each of

[7]A vector x is said to be L_1 normalized if $\sum_i X_i = 1$.
[8]A vector x is said to be L_2 normalized if $\sum_i X_i^2 = 1$.

Algorithm 5 MinMax criterion for new class creation.

1: **function MinMaxCriterion**($[P(C_1|x) \ \ldots \ P(C_k|x)]$):
2: **Input**: $[P(C_1|x) \ldots P(C_k|x)]$ probability distribution of existing classes for a data point x
3: **Output:** *decision* : **true** iff new class needs to be created
4: k is the current number of classes
5: $maxProb$ = max($P(C_j|x)$); $minProb$ = min($P(C_j|x)$)
6: **if** $\frac{maxProb}{minProb} < 2$ **then**
7: *decision* = true
8: **end if**
9: **end function**

these is a possible implementation of the "nearUniform" subroutine of Algorithm 3. As noted above, the intuition is that new classes should be introduced to hold for x when the probabilities of x belonging to existing classes are close to uniform. In the JS criterion, we require that Jensen-Shanon divergence[9] between the posterior class distribution for x to the uniform distribution be less than $\frac{1}{k}$. The MinMax criterion is a somewhat simpler approximation to this intuition: a new cluster is introduced if the maximum probability is no more than twice the minimum probability.

4.2 Experimental Results: Entity Clustering and Document Classification

We now seek to experimentally answer the questions raised in the introduction. How robust are existing SSL methods, if they are given incorrect information about the number of classes present in the data? How well they perform if we have seed data for only some of the classes? Do the exploratory versions of the SSL methods perform better? How does Exploratory EM compare with the existing methods like Gibbs sampling with Chinese Restaurant Process (CRP)?

We used three publicly available datasets for our experiments. The statistics are summarized in Table 4.1. The first dataset is the Delicious_Sports dataset [126]. This is an entity classification dataset, which contains items extracted from 57K HTML tables in the sports domain (from pages that had been tagged by the social bookmarking system del.icio.us). The features of an entity are ids for the HTML table columns in which it appears. We used only the Delicious_Sports dataset from the WebSets corpora, because we wanted a fully labeled dataset and one that does not violate our assumption that each entity belongs to only class. This dataset contains 282 labeled entities described by 721 features and 26 non-overlapping classes (e.g., "NFL teams", "Cricket teams"). The second is the widely-used 20-Newsgroups dataset [109]. We used the "bydate" dataset, which contains total of 18,774 text documents, with vocabulary size of 61,188. There are 20 non-overlapping classes and the entire dataset is labeled. The third dataset is the Reuters-21578 dataset published by Cai et al. [20]. This corpus originally contained 21,578 documents from 135 overlapping categories. Cai et al. discarded documents with multiple category labels, resulting in 8,293 documents (vocabulary size=18,933) in 65 non-overlapping categories.

[9]The Jensen-Shannon divergence between p and q is the average Kullback-Leiber divergence of p and q to a, the average of p and q, i.e., $\frac{1}{2}(KL(p||a + KL(q||a))$.

Table 4.1: Datasets used in this chapter.

Dataset	# Datapoints	# Features	# Classes
Delicious_Sports	282	721	26
Reuters	8,293	18,933	65
20-Newsgroups	18,774	61,188	20

Table 4.2: Comparison of Exploratory EM w.r.t. EM for different datasets and class creation criteria. For each exploratory method we report the macro avg. F1 over seed classes followed by avg number of clusters generated. e.g., For 20-Newsgroups dataset, Exploratory EM with K-Means and MinMax results in 57.4 F1 and generates 22 clusters on avg. ▲ (and △) indicates that improvements are statistically significant w.r.t EM with 0.05 (and 0.1) significance level.

Dataset (#seed / #total classes)	Language Model	EM (non-exploratory)	Exploratory EM			EM with Best m extra classes
			Random	MinMax	JS	
Delicious_Sports (5/26)	KM	60.9	84.8 (55) ▲	89.5 (30) ▲	**90.6** (46) ▲	69.4 (10) ▲
	NB	46.3	67.8 (38) ▲	45.4 (06)	**88.4** (51) ▲	65.8 (10) ▲
	VMF	64.3	66.7 (06)	72.8 (06) △	63.0 (06)	**78.2** (09) ▲
20-Newsgroups (6/20)	KM	44.9	53.0 (22) ▲	**57.4** (22) ▲	39.4 (99) ▼	49.8 (11) ▲
	NB	34.0	34.0 (06)	34.6 (07)	34.0 (06)	**35.0** (07)
	VMF	18.2	18.2 (06)	09.5 (09) ▼	19.8 (06)	**20.3** (10) ▲
Reuters (10/65)	KM	8.9	13.7 (19) ▲	12.0 (16) △	**27.4** (100) ▲	16.3 (14) ▲
	NB	6.4	10.6 (10)	10.4 (10)	**18.5** (077) ▲	16.1 (15)
	VMF	10.5	10.5 (10)	20.7 (11) ▲	**30.4** (062) ▲	20.6 (16) △

Table 4.2 shows the performance of seeded K-Means, seeded Naive Bayes, and seeded vMF using 5 different algorithms. For each dataset only a few of the classes present in the data (5 for Delicious_Sports, and 6 for 20-Newsgroups and 10 for Reuters), are given as seed classes to all the algorithms. Five percent of the datapoints were given as training data for each "seeded" class. The first method, shown in the column labeled EM, uses these methods as conventional SSL learners. The second method is Exploratory EM with the Random criterion i.e., introducing new classes for unlabeled datapoints uniformly at random with a predefined value of new class creation probability. The third method is Exploratory EM with the simple MinMax new-class introduction criterion, and the forth is Exploratory EM with the JS criterion. The last one is an upper bound on the performance of EM with m extra classes, computed by varying the value of m and picking the best one for each row in Table 4.2.

Exploratory EM performs hard clustering of the dataset, i.e., each datapoint belongs to only one cluster. For all methods, for each cluster we assign a label that maximizes accuracy (i.e., majority label for the cluster). Thus using a complete set of labels we can generate a single label per datapoint. Further even if Exploratory EM produces more number of clusters than the actual number of classes present in the datasets, they get mapped to the actual classes, hence the confusion matrices have both number of dimensions equal to the actual number of classes. The reported F1 value is computed by macro averaging the F1 values of seed classes only. Note that, for a given dataset, number of seed classes, and training percentage per seed class, there are many ways to generate a train-test partition. We report results using 10 random train-test partitions of each dataset. The same partitions are used to run all the algorithms being compared

and to compute the statistical significance of the results.

We first consider the value of exploratory learning. With the JS criterion, the exploratory extension gives comparable or improved performance on 8 out of 9 cases. In 5 out of 8 cases the gains are statistically significant. With the simpler MinMax criterion, the exploratory extension results in performance improvements in 7 out of 9 cases, and significantly reduces performance only in one case. The number of classes finally introduced by the MinMax criterion is generally smaller than those introduced by JS criterion.

For both SSL and exploratory systems, the seeded K-Means method gives good results on all 3 datasets. In our MATLAB implementation, the running time of Exploratory EM is longer, but not unreasonably so: on average for the 20-Newsgroups dataset Semisup-KMeans took 95 sec. while Explore-KMeans took 195 sec. and for the Reuters dataset, Semisup-KMeans took 7 sec. while Explore-KMeans took 28 sec. These results are summarized in Table 4.3.

Dataset	Avg. runtime of EM in sec.	Average run-time of Exploratory EM in multiple of EM
Delicious_Sports	0.2	2.5
20-Newsgroups	95.0	2.0
Reuters	7.0	4.0

Table 4.3: Comparison of average runtimes of Exploratory EM w.r.t EM.

We can also see that Random criterion shows significant improvements over the baseline EM method in 4 out of 9 cases. Note that this method is kind of supervised because the new class creation probability is learnt for each dataset by using our JS criterion first. After running the Exploratory EM method with JS criterion, the new class creation probability is computed as follows: $P_{new} = (\#\text{newly created classes})/(\#\text{unlabeled datapoints})$. For each unlabeled datapoint the Random criterion induces a new class with probability P_{new}. Exploratory EM method with MinMax and JS criterion shows significant improvements in 5 out of 9 cases. In terms of magnitude of improvements, JS is superior to Random.

Next we compare Exploratory EM with another baseline named "EM with m extra classes". The last column of Table 4.2 shows the best performance of this baseline by varying m = {0, 1, 2, 5, 10, 20, 40}, and choosing that value of m for which seed class F1 is maximum. Since the "best m extra classes" baseline is making use of the test labels to pick right number of classes, it cannot be used in practice; however Exploratory EM methods produce comparable or better performance with this strong baseline.

To better understand the qualitative behavior of our methods, we conducted some further experiments with Semisup-KMeans with the MinMax criterion. We constructed confusion matrices for the classification task, to check how different methods perform on each dataset.[10] Figure 4.1 (a) shows the confusion matrices for EM (top row) and Exploratory EM (bottom row) with five and fifteen seeded classes. We can see that EM with only five seed classes clearly confuses the unexpected classes with the seed classes, while Exploratory EM gives better quality results. Having seeds for more classes helps both EM and Exploratory EM, but EM still tends to confuse the unexpected classes with the seed classes. Figure 4.1 (b) shows similar results on the

[10]For purposes of visualization, introduced classes were aligned optimally with the true classes.

20-Newsgroups dataset, but shows the confusion matrix after 1 iteration and after 15 iterations of EM. It shows that EM after 15 iterations has made limited progress in improving its classifier when compared to Exploratory EM.

Finally, we compare the two class creation criteria, and show a somewhat larger range of seeded classes, ranging from 5 to 15 (out of 20 actual classes). In Figure 4.2 each of the confusion-matrices is annotated with the strategy, the number of seed classes and the number of classes produced. (For example, plot "MinMax-C5(23)" describes Explore-KMeans with MinMax criterion and 5 seed classes which produces 23 clusters.) We can see that MinMax criterion usually produces a more reasonable number of clusters, closer to the ideal value of 20; however, the performance of the JS method in terms of seed class accuracy is comparable to the MinMax method.

These trends are also shown quantitatively in Figure 4.3, which shows the result of varying the number of seeded classes (with five seeds per class) for Explore-KMeans and Semisup-KMeans; the top shows the effect on F1, and the bottom shows the effect on the number of classes produced (for Explore-KMeans only). Figure 4.4 shows a similar effect on the Delicious_Sports dataset: here we systematically vary the number of seeded classes (using 5 seeds per seeded class, on the top), and also vary the number of seeds per class (using 10 seeded classes, on the bottom.) The left-hand side compares the F1 for Semisup-KMeans and Explore-KMeans, and the right-hand side shows the number of classes produced by Explore-KMeans. For all parameter settings, Explore-KMeans is better than or comparable to Semisup-KMeans in terms of F1 on seed classes.

4.3 Evaluation on Non-seed Classes

Table 4.4 presents the results for macro-averaged F1 scores on seeded and unanticipated classes. Best scores in each row are bold-faced. We can see that on a relatively easier dataset like Delicious_Sports which has clearly separable classes, our methods could easily discover good quality extra clusters corresponding to unanticipated classes. However, as the complexity of classification task increases, our methods prove to be less effective. E.g., Reuters dataset has less clearly separable classes, and only small fraction of them are given as input to the algorithm. For this task Exploratory EM could induce many new clusters but they do not directly correspond to the ideal class definitions in the test data resulting in low F1 scores on unanticipated classes.

Further the Exploratory EM method resulted in better F1 score on unanticipated classes when compared to EM with best m-extra classes. Thus we observe from Tables 4.2 and 4.4 that dynamically adding new clusters of datapoints that do not belong to any of the known classes does reduce semantic drift of existing classes even if the newly added clusters are not meaningful (the newly added clusters might not have 1 to 1 correspondence with the unanticipated classes).

4.4 Additional Results

We later created four entity classification datasets using subsets of NELL ontology (see Figure 7.1) and generated feature vectors using occurrences with text patterns and HTML table columns in the ClueWeb09 corpus. Description of these four datasets can be found in Section 7.4. We

Dataset (#seed / #total classes)	Model	Exploratory EM				EM with best m-extra classes	
		MinMax		JS			
		Seed	Unanticipated	Seed	Unanticipated	Seed	Unanticipated
Delicious_Sports (5/26)	KM	90.8	86.0 (25)	90.6	**92.3 (41)**	65.7	5.8 (10)
	NB	46.9	7.1 (01)	88.4	**93.1 (46)**	58.4	7.7 (10)
	VMF	68.4	**8.1 (01)**	63.0	7.7 (01)	66.6	7.4 (10)
20-Newsgroups (6/20)	KM	57.4	**25.1 (16)**	39.4	24.2 (93)	48.5	5.0 (10)
	NB	34.6	3.1 (02)	30.0	2.5 (00)	29.0	**4.2 (10)**
	VMF	9.5	11.4 (03)	19.8	**15.6 (00)**	18.5	4.1 (10)
Reuters (10/65)	KM	12.0	4.8 (06)	27.4	**10 (90)**	11.4	3.1 (20)
	NB	10.4	2.5 (00)	15.6	2.4 (67)	15.5	**2.6 (20)**
	VMF	20.7	4.8 (01)	30.4	**5.8 (52)**	18.9	5.1 (20)

Table 4.4: Evaluation of Exploratory EM in terms of macro-averaged F1 on non-seed classes. For each method we present the F1 score on unanticipated classes followed by number of extra clusters added by the algorithm. E.g., KM model with MinMax criterion when applied on Delicious_Sports dataset added 25 extra clusters, with 86% F1 score on unanticipated classes.

present the results of EM and Exploratory EM methods on these more recent datasets to check whether we observe trends similar to Section 4.2.

Dataset	#Seed/#Ideal classes	Level	Macro-avg. seed class F1	
			EM	Exploratory EM
Text-Small	2/3	2	46.6	**64.4** ▲
	4/7	3	23.5	**32.7** ▲
Text-Medium	3.9/4	2	**53.2**	50.2 ▼
	9.4/24	3	**27.9**	27.0
	2.4/10	4	17.4	**25.8** ▲
Table-Small	2/3	2	69.5	**75.8** ▲
	4/7	3	36.8	**43.9** ▲
Table-Medium	3.9/4	2	**62.7**	61.3
	9.4/24	3	43.7	**49.1** ▲
	2.4/10	4	47.3	**52.2** △

Table 4.5: Comparison of semi-supervised vs. exploratory learning using KM representation on NELL's entity classification datasets (described in Section 7.4).

In Table 4.5, we present the performance of traditional semi-supervised EM and compare it against Exploratory EM at different levels of class hierarchy. Best performance in each row is bold faced. We can see that in 7 out of 10 cases, Explore-KMeans outperforms Semisup-KMeans in terms of macro-averaged seed class F1. Also note that for 3 out of 4 datasets, the improvements are more for the refined levels of hierarchy (the fraction of unanticipated classes increases with the level of refinement). Runtimes of these methods are compared in Table 4.6. We can see that average runtime of Exploratory EM method is higher than that of EM, however

not unreasonably so.

Dataset	Avg. runtime of EM in sec.	Average run-time of Exploratory EM in multiple of FLAT-EM
Text-Small	53	8
Text-Medium	524	5
Table-Small	50	3
Table-Medium	5932	4

Table 4.6: Comparison of average runtimes of Exploratory EM w.r.t EM.

4.5 Experiments with Chinese Restaurant Process

The Exploratory EM method is broadly similar to non-parametric Bayesian methods, such as the Chinese Restaurant Process (CRP) [11]. CRP is often used in non-parametric models (e.g., topic models) that are based on Gibbs sampling, and indeed, since it is straightforward to replace EM with Gibbs-sampling, one can use this approach to estimate the parameters of any of the models considered here (i.e., multinomial Naive Bayes, K-Means, and the von Mises-Fisher distribution). Algorithm 6 presents a seeded version of a Gibbs sampler based on this idea. In brief, Algorithm 6 starts with a classifier trained on the labeled data. Collapsed Gibbs sampling is then performed over the latent labels of unlabeled data, incorporating the CRP into the Gibbs sampling to introduce new classes. (In fact, we use block sampling for these variables, to make the method more similar to the EM variants.)

The subroutine CRPPick is explained in Algorithm 7. This function concatenates the posterior distribution of classes given a datapoint with the new class creation probability or concentration parameter of CRP, renormalizes this vector and then samples a label from this new distribution. If the new class is selected then the number of classes is incremented by one.

Note that this algorithm is naturally "exploratory", in our sense, as it can produce a number of classes larger than the number of classes for which seed labels exist. However, unlike our exploratory EM variants, the introduction of new classes is not driven by examples that are "hard to classify"—i.e., have nearly-uniform posterior probability of membership in existing classes. In the CRP method, the probability of creating a new class depends on the data point, but it does not explicitly favor cases where the posterior over existing classes is nearly uniform.

To address this issue, we also implemented a variant of the seeded Gibbs sampler with CRP, in which the examples with nearly-uniform distributions are more likely to be assigned to new classes. This variant is shown in Algorithm 8, which replaces the routine CRPPick in the Gibbs sampler—in brief, we simply scale down the probability of creating a new class by the Jensen-Shannon divergence of the posterior class distribution for x to the uniform distribution. Hence the probability of creating a new class explicitly depends on how well the given data point fits in one of the existing classes.

Next we compare the performance of CRPGibbs with Explore-KMeans and Semisup-KMeans. We consider two versions of CRP-Gibbs, one using the standard CRP and one using our proposed modified CRP criterion for new class creation that is sensitive to the near-uniformity of instance's

Algorithm 6 Exploratory Gibbs Sampling with Chinese Restaurant Process.

function GibbsCRP (X^l, Y^l, X^u, $\{C_1 \dots C_k\}$) : $C_{k+1} \dots C_m$, Y^u
Input: X^l labeled data points; Y^l labels of X^l; X^u unlabeled data points;
$\{C_1 \dots C_k\}$ set of known classes x's belong to; P_{new} probability of creating a new class.
Output: $C_{k+1} \dots C_m$ newly-discovered classes; Y^u labels for X^u
for x in X^u **do**
    Save a random class from $\{C_1 \dots C_k\}$ for x in Y^u
end for
Set $m = k$
for t : 1 to $numEpochs$ **do**
    for X_i in X^u **do**
        Let y_i's be X_i's label in epoch $t-1$
        predict $P(C_j|X_i, Y^l \cup Y^u - \{y_i\})$ for all labels $1 \leq j \leq m$
        y_i', m' = CRPPick(P_{new}, $P(C_1|X_i), \dots, P(C_{m+1}|X_i)$)
        Save y_i' as X_i's label in epoch t
        $m = m'$
    end for
end for
end function

posterior class distribution. CRP-Gibbs uses the same instance representation as our K-Means variants, i.e., L_1 normalized TFIDF features.

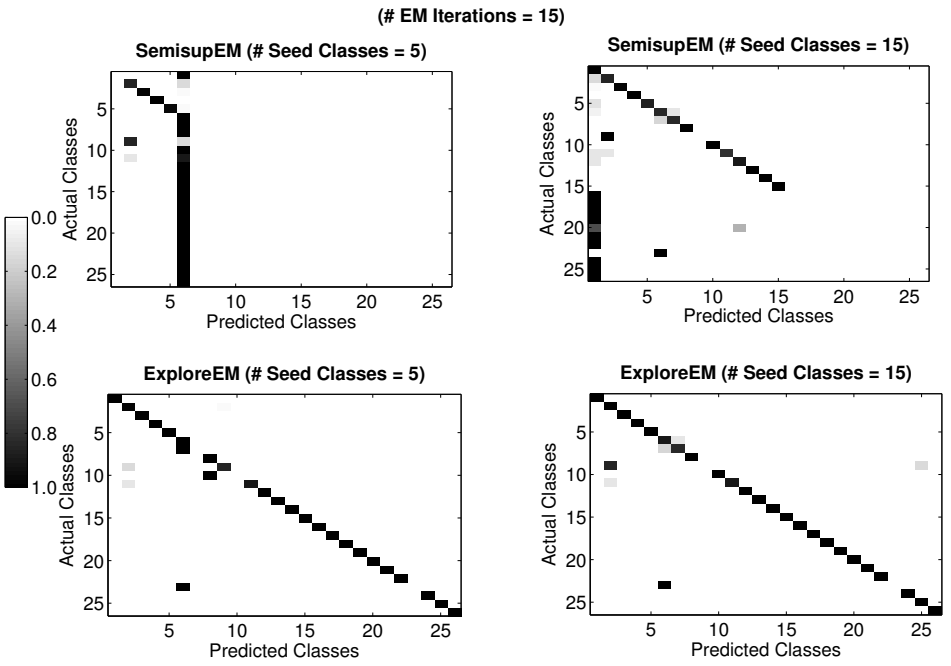
It is well-known that CRP is sensitive to the concentration parameter P_{new}. Figure 4.5 (a) and (b) shows the performance of all the exploratory methods, as well as Semisup-KMeans, as the concentration parameter is varied from 10^{-8} to 10^{-2}. (For Explore-KMeans and Semisup-KMeans methods, this parameter is irrelevant). We show F1, the number of classes produced, and run-time (which is closely related to the number of classes produced). The results show that a well-tuned seeded CRP-Gibbs can obtain good F1-performance, but at the cost of introducing many unnecessary clusters. The modified Explore-CRP-Gibbs performs consistently better, but not better than Explore-KMeans, and Semisup-KMeans performs the worst.

Algorithm 7 CRP criterion for new class creation.

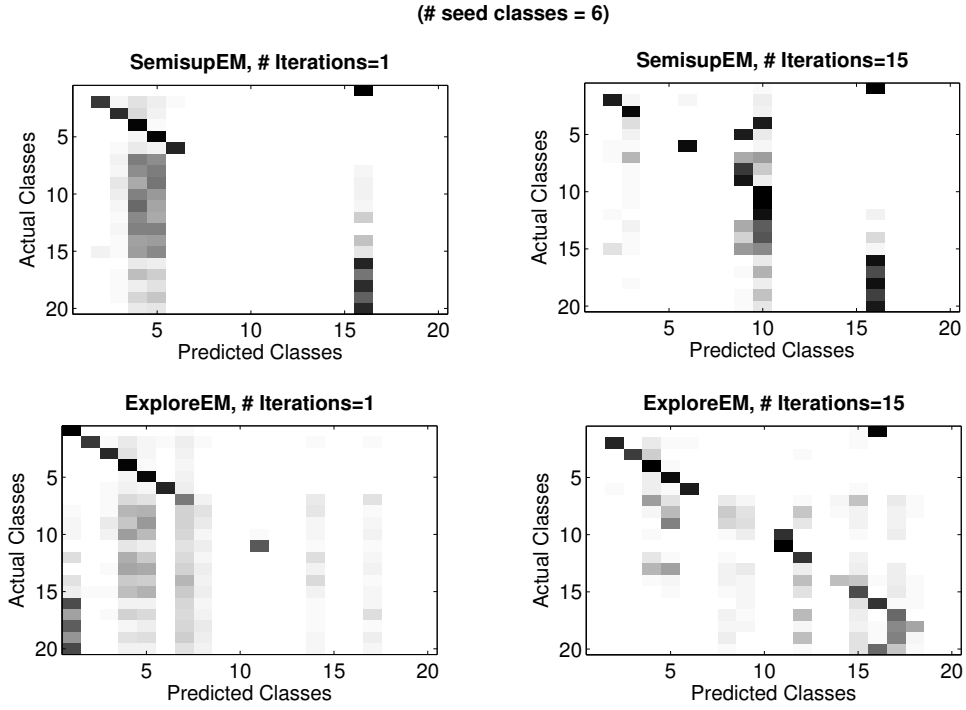
function CRPPick (P_{new}, $P(C_1|x), \dots, P(C_k|x)$) : y, k'
Input: P_{new} probability of creating a new class;
$P(C_1|x), \dots, P(C_k|x)$ probability of existing classes given x
Output: y class for x; k' new number of classes
$P_{all} = [P(C_1|x) \dots P(C_k|x) P_{new}]$
Normalize P_{all} such that its L_1 norm equals 1.
{Sample a class using P_{all}}
y = sample from distribution P_{all}
if $y == k+1$ **then**
    $k' = k+1$
end if
end function

Algorithm 8 Modified CRP criterion for new class creation.

function ModCRPPick (P_{new}, $P(C_1|x), \ldots, P(C_k|x)) : y, k'$
Input: P_{new} probability of creating a new class;
$P(C_1|x), \ldots, P(C_k|x)$ probability of existing classes given x
Output: y class for x; k' new number of classes
u = $[1/k \ldots 1/k]$ {uniform distribution with k classes}
d = Jensen-Shannon-Divergence(u, $P(C_j|x)$)
q = $\frac{P_{new}}{((k)*d)}$
if a coin with bias q is heads **then**
    {create a new class and assign to that}
    $y = k + 1$ and $k' = k + 1$
else
    {assign to an existing class}
    $k' = k$ and y = sample from distribution $[P(C_1|x) \ \ldots \ P(C_k|x)]$
end if
end function



(a) Confusion matrices, varying # seed classes, for the Delicious_Sports dataset



(b) Confusion matrices, varying # EM iterations for the 20-Newsgroups dataset.

Figure 4.1: Comparison of EM (SemisupEM) vs. Exploratory EM (ExploreEM) using the Min-Max criterion and K-Means model.

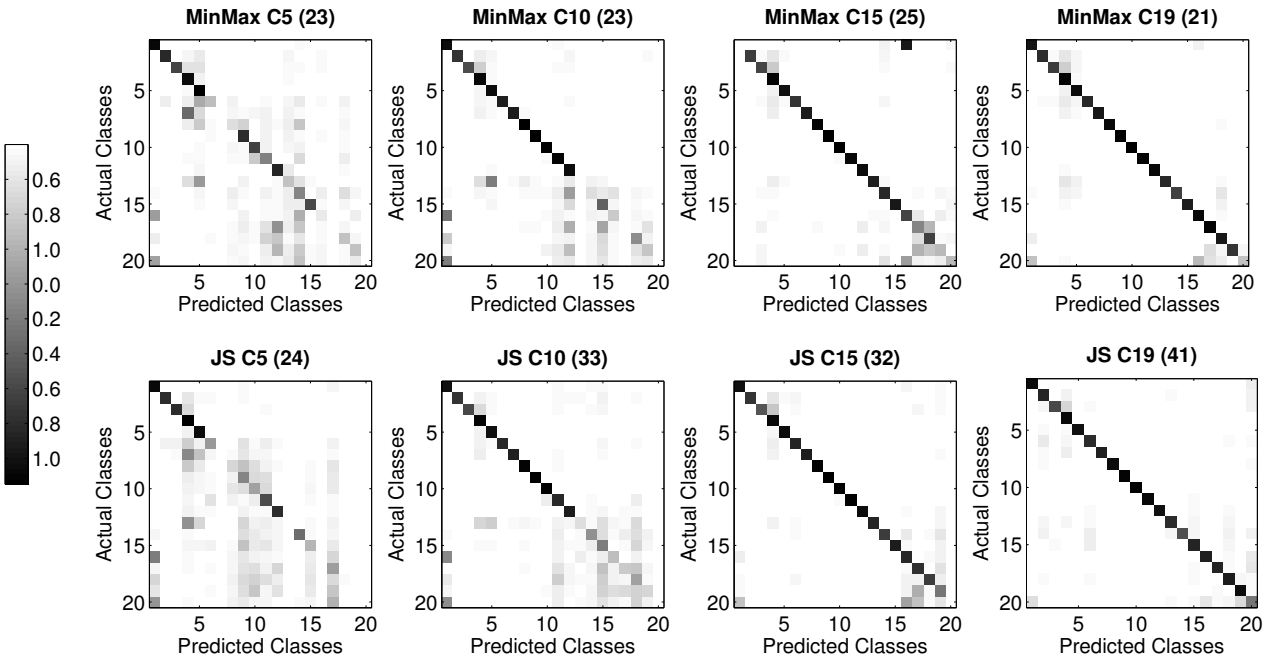


Figure 4.2: 20-Newsgroups dataset : Comparison of MinMax vs. JS criterion for Exploratory EM.

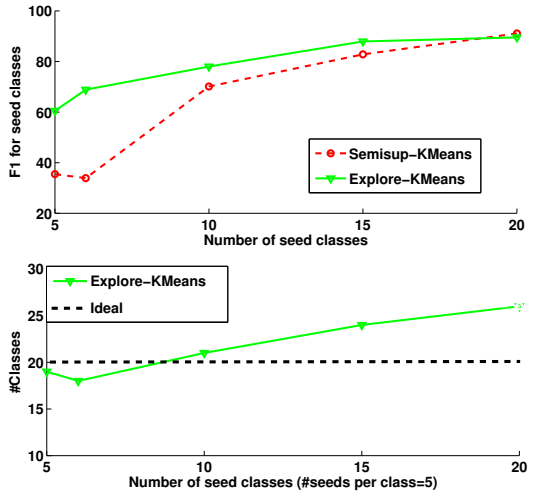


Figure 4.3: 20-Newsgroups dataset: varying the number of seed classes using the MinMax criterion (Explore-KMeans refers to Exploratory EM with K-Means model).

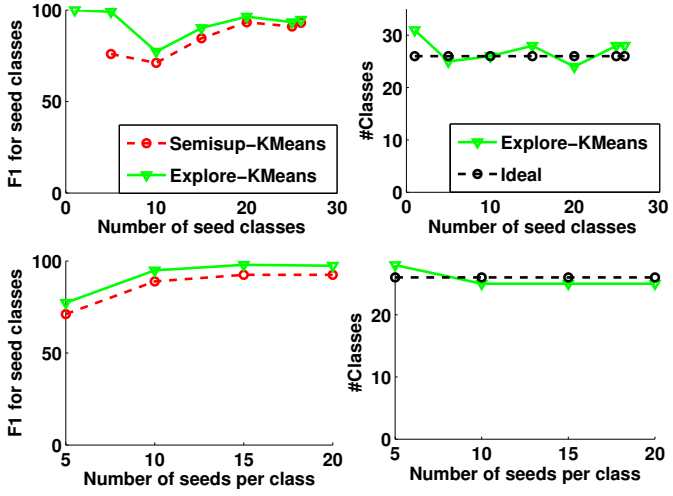
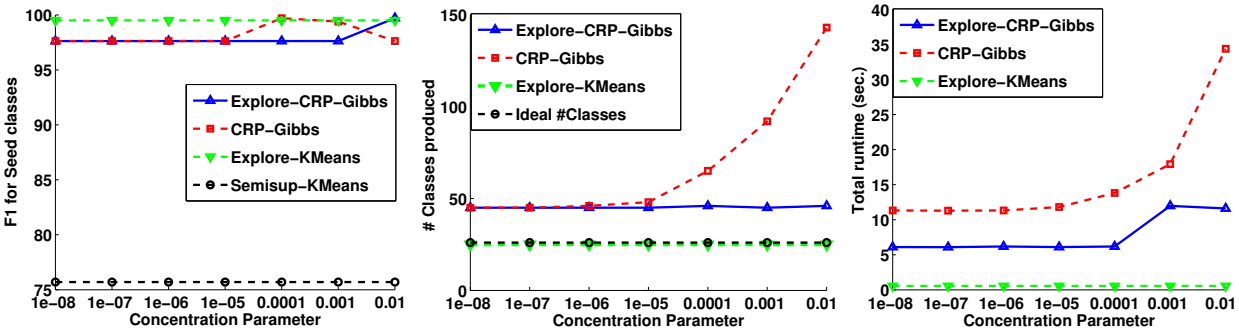
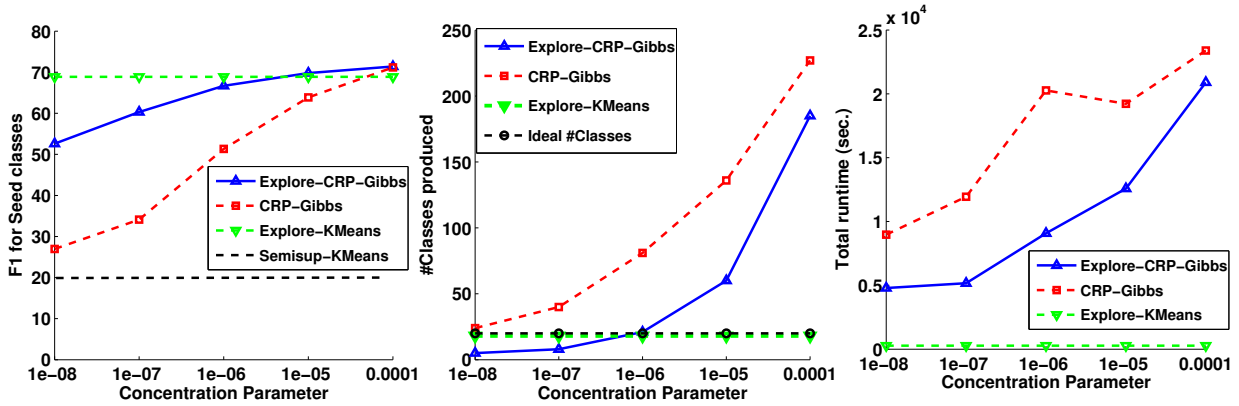


Figure 4.4: Delicious Sports dataset: Top, varying the number of seed classes (with five seeds per class). Bottom, varying the number of seeds per class (with 10 seed classes). Semisup-KMeans refers to Semi-EM algorithm and Explore-KMeans refers to Exploratory EM with K-Means model.



(a) Delicious_Sports dataset, #seed classes=5



(b) 20-Newsgroups dataset, #seed classes=6

Figure 4.5: Comparison of Exploratory EM w.r.t. CRPGibbs by varying the concentration parameter of CRP.

4.6 Related Work: Semi-supervised Learning, Negative Class Discovery, Topic Detection and Tracking

In this chapter we described and evaluated a novel multiclass SSL method "Exploratory EM" that is more robust when there are unanticipated classes in the data—or equivalently, when the algorithm is given seeds from only some of the classes present in the data. To the best of our knowledge this specific problem has not been explored in detail before, even though in real-world settings, there can be unanticipated (and hence unseeded) classes in any sufficiently large-scale multiclass SSL task.

More generally, however, it has been noted before that SSL may suffer due to the presence of unexpected structure in the data. For instance, Nigam et al's early work on SSL based EM with multinomial Naive Bayes [98] noted that adding too much unlabeled data sometimes hurt performance on SSL tasks, and discusses several reasons this might occur, including the possibility that there might not be a one-to-one correspondence between the natural mixture components (clusters) and the classes. To address this problem, they considered modeling the positive class with one component, and the negative class with a mixture of components. They propose to choose the number of such components by cross-validation; however, this approach is relatively expensive, and inappropriate when there are only a small number of labeled examples (which is a typical case in SSL). More recently, McIntosh [87] described heuristics for introducing new "negative categories" in lexicon bootstrapping, based on a domain-specific heuristic for detecting semantic drift with distributional similarity metrics. Our setting is broadly similar to these works, except that we consider this task in a general multiclass-learning setting, and do not assume seeds from the explicitly-labeled "negative" class, which is a mixture; instead, we assume seeds from known classes only. Thus we assume that the data fits a mixture model with a one-to-one correspondence with the classes, but only after the learner introduces new classes hidden in the data. We also explore this issue in much more depth experimentally, by systematically considering the impact of having too few seed classes, and propose and evaluate a solution to the problem. There has also been substantial work in the past to automatically decide the right "number of clusters" in unsupervised learning [30, 40, 59, 88, 102, 138]. Many of these techniques are built around K-Means and involve running it multiple times for different values of K. Exploratory learning differs in that we focus on a SSL setting, and evaluate specifically the performance difference on the seeded classes, rather than overall performance differences.

There is also a substantial body of work on constrained clustering; for instance, Wagstaff et al [133] describe a constrained clustering variant of K-Means "must-link" and "cannot-link" constraints between pairs. This technique changes the cluster assignment phase of K-Means algorithm by assigning each example to the closest cluster such that none of the constraints are violated. SSL in general can be viewed as a special case of constrained clustering, as seed labels can be viewed as constraints on the clusters; hence exploratory learning can be viewed as a subtype of constrained clustering, as well as a generalization of SSL. However, our approach is different in the sense that there are more efficient methods for dealing with seeds than arbitrary constraints.

In this section we focus on EM-like SSL methods. Another widely-used approach to SSL is label propagation. In the modified adsorption algorithm [120], one such graph-based label

propagation method, each datapoint can be marked with one or more known labels, or a special "dummy label" meaning "none of the above". Exploratory learning is an extension that applies to a different class of SSL methods, and has some advantages over label propagation: for instance, it can be used for inductive tasks, not only transductive tasks. Exploratory EM also provides more information by introducing multiple "dummy labels" which describe multiple new classes in the data.

A third approach to SSL involves unsupervised dimensionality reduction followed by supervised learning (e.g., [122, 125]). Although we have not explored their combination, these techniques are potentially complementary with exploratory learning, as one could also apply EM-like methods in a lower-dimensional space (as is typically done in spectral clustering). If this approach were followed then an exploratory learning method like Exploratory EM could be used to introduce new classes, and potentially gain better performance, in a semi-supervised spectral setting.

One of our benchmark tasks, entity classification, is inspired by the NELL (Never Ending Language Learning) system [25]. NELL performs broad-scale multiclass SSL. One subproject within NELL [92] uses a clustering technique for discovering new relations between existing noun categories—relations not defined by the existing hand-defined ontology. Exploratory learning addresses the same problem, but integrates the introduction of new classes into the SSL process. Another line of research considers the problem of "open information extraction", in which no classes or seeds are used at all [41, 126, 145]. Exploratory learning, in contrast, can exploit existing information about classes of interest and seed labels to improve performance.

Another related area of research is novelty detection. Topic detection and tracking task aims to detect novel documents at time t by comparing them to all documents till time $t-1$ and detects novel topics. Kasiviswanathan et al. [70] assumes the number of novel topics is given as input to the algorithm. Masud et al. [85] develop techniques on streaming data to predict whether next data chunk at time $t+1$ is novel or not based on data stream observed till time t. Our focus is on improving performance of semi-supervised learning when the number of new classes is unknown. Bouveyron [14] worked on the EM approach to model unknown classes, but the entire EM algorithm is run for multiple numbers of classes; we use this as an upper bound on the baseline described as 'EM with best m extra classes' in Table 4.2. Our algorithm jointly learns labels as well as new classes. Schölkopf et al. [112] defines a problem of learning a function over the data space that isolates outliers from class instances. Our approach is different in the sense we do not focus on detecting outliers for each class but on detecting new and unanticipated classes.

Because Exploratory EM is broadly similar to non-parametric Bayesian approaches, we also compared Explore-KMeans to a seeded version of an unsupervised mixture learner that explores differing numbers of mixture components with the Chinese Restaurant process (CRP). Explore-KMeans is faster than this approach, and more accurate as well, unless the parameters of the CRP are very carefully tuned. Explore-KMeans also generates a model that is more compact, having close to the true number of clusters. The seeded CRP process can be improved, moreover, by adapting some of the intuitions of Explore-KMeans, in particular by introducing new clusters most frequently for hard-to-classify examples (those with nearly-uniform posteriors).

4.7 Conclusion

In this chapter, we investigated and improved the robustness of SSL methods in a setting in which seeds are available for only a subset of the classes—the subset of most interest to the end user. We performed systematic experiments on fully-labeled multiclass problems, in which the number of classes is known. We showed that if a user provides seeds for only some, but not all, classes, then SSL performance is degraded for several popular EM-like SSL methods (semi-supervised multinomial Naive Bayes, seeded K-Means, and a seeded version of mixtures of von Mises-Fisher distributions). We then described a novel extension of the EM framework called Exploratory EM, which makes these methods much more robust to unseeded classes. Exploratory EM introduces new classes on-the-fly during learning based on the intuition that hard-to-classify examples—specifically, examples with a nearly-uniform posterior class distribution—should be assigned to new classes. The exploratory versions of these SSL methods often obtained dramatically better performance—e.g., on Delicious_Sports dataset up to 49% relative improvements in F1, on 20-Newsgroups dataset up to 28% relative improvements in F1, and on Reuters dataset up to 190% relative improvements in F1.

4.8 Future Research Directions

The techniques we developed till now are limited to flat classification tasks. In the real world data, there are a variety of constraints between classes like inclusion, mutual exclusion etc. In the next few chapters, we extend the Exploratory Learning technique to model arbitrary class constraints. Further, information about entities can be present in multiple views, it is an interesting research direction to extend SSL methods to handle multi-view data. The upcoming chapters talk about these and other directions of proposed research.

Further, the exploratory learning techniques we described here are limited to problems where each data point belongs to only one class. An interesting direction for future research can be to develop such techniques for multi-label classification, and hierarchical classification. Another direction can be create more scalable parallel versions of Explore-KMeans for much larger datasets, e.g., large-scale entity-clustering tasks.

Chapter 5

Constrained Semi-supervised Learning with Multi-view Datasets

In semi-supervised learning, sometimes the information about datapoints is present in multiple views. In this chapter we propose an optimization based method to tackle semi-supervised learning in the presence of multiple views. Our techniques make use of mixed integer linear programming formulations along with the EM framework to find consistent class assignments given the scores in each data view. We compare our techniques against existing baselines, on a number of multi-view datasets. Our proposed techniques give state-of-the-art performance in terms of F1 score, outperforming a well-studied SSL method based on co-training. Later, in Chapter 6, we show that our techniques can be easily extended to multi-view learning in the presence of hierarchical class constraints. These extensions improve the macro-averaged F1 score on a hierarchical multi-view dataset. This work is currently in preparation for submission [123].

5.1 Motivation

In multiclass semi-supervised learning, sometimes the information about datapoints is present in multiple views. For instance consider Web document classification: a learning algorithm can use two views, the text within the document and the anchor texts of its inbound hyper-links. Similarly, in an information extraction task to populate a Knowledge Base (KB) like NELL [25], each noun-phrase to be classified has different sets of features or data views associated with it; e.g., text contexts that appeared with it, its occurrences in HTML table-columns, morphological features, and so on. As an example, consider the two view dataset in Figure 5.1, with each noun-phrase being represented by distributional features w.r.t its occurrences with text-patterns and in HTML-Tables. For the noun-phrase "Carnegie Mellon University", a text-pattern feature ("_arg1 is located in", 100) denotes that the noun-phrase "Carnegie Mellon University" appeared in arg1 position of the context "_arg1 is located in" 100 times in the input corpus of sentences. For the same noun-phrase having a HTML-Table feature ("doc04::2:1", 1) means that the noun-phrase "Carnegie Mellon University" appeared once in HTML table 2, column 1, from document id "doc04".

A common approach to multi-view learning is to concatenate the feature vectors for each

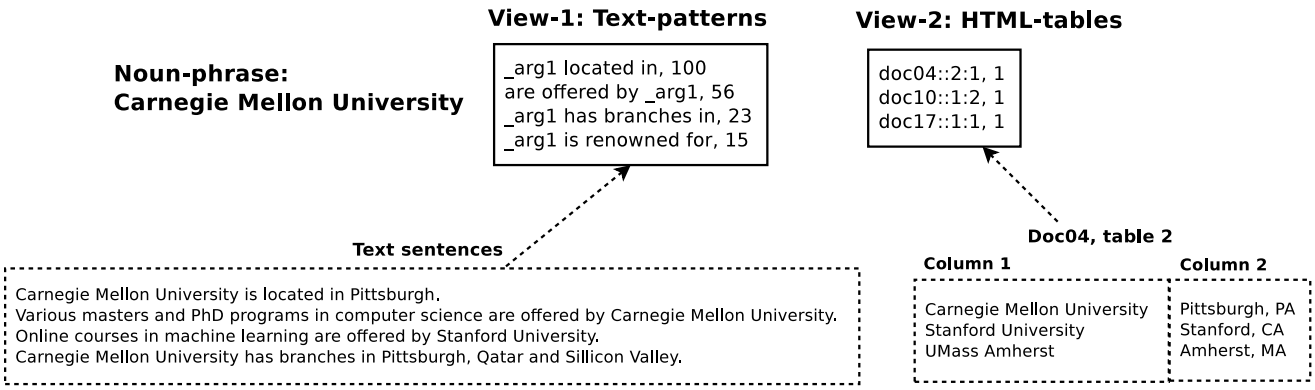


Figure 5.1: An example of multi-view dataset for Knowledge Base population task. For each noun-phrase distributional features are derived from two data sources. Occurrences of the noun-phrase with text-patterns result in View-1 and occurrences of the noun phrase in HTML table columns result in View-2.

view but this is not always optimal. Multi-view learning [13, 97, 122, 131, 144] addresses this problem by introducing a different function to model each view, and jointly optimizing all the functions to exploit the redundant views and improve learning performance.

Multiple views are only one issue arising in complex real-world learning tasks. For instance, in the above mentioned KB population task, the labels assigned to each noun-phrase need to be consistent with the hierarchical class constraints posed by the KB ontology. We deal with hierarchical multi-view learning problems in Chapter 6.

There has already been a lot of research in the individual areas of multi-view learning [13, 97, 122, 131], semi-supervised learning [25, 98, 128], and learning in the presence of class hierarchies [54, 67, 95, 127]. However, the problem of unifying these aspects into a single framework is less explored. In this chapter, we focus on the problem of multi-view semi-supervised learning and later extend it to incorporate class hierarchies in Chapter 6. We propose an optimization based formulation for this problem in the EM framework, which extends the popular K-Means algorithm. The main idea behind our approach is to use different binary labels to model membership in each view. We then use mixed integer programming formulations to optimally assign a set of labels to each instance. Instead of collectively optimizing labels for all datapoints, we solve an integer linear program for every datapoint, deciding the optimal label assignments for the datapoint. We compare our methods against a state-of-the-art co-training approach, applied to the semi-supervised spherical K-Means algorithm [10].

Our experiments show that our proposed methods work as well as or better than standard multi-view baselines like multiplication (or addition) of scores, concatenation of feature vectors, and outperform a well-studied co-training based baseline, when applied in the standard multi-view k-way classification setting.

In Section 5.2 we propose our optimization framework for multi-view semi-supervised learning. We then discuss various multi-view methods that can be instantiated using this framework. We then describe the datasets and experimental methodology in Section 5.3, and present our findings in Section 5.4. Finally, we discuss related work in Section 5.5, followed by conclusions.

5.2 Multi-View Semi-supervised Learning

The spherical K-Means algorithm [37], i.e., the K-Means algorithm with cosine similarity, is a popular method for clustering high-dimensional data such as text. In this section we will first review this existing algorithm, we then present our proposed extension called "multi-view spherical K-Means". There are different ways in which scores from multiple views can be combined to infer labels of every data point. We formulate the multi-view semi-supervised K-Means as a generic optimization problem, and instantiate various existing methods, as well as our proposed methods of combining scores from multiple views as special cases of this optimization problem.

5.2.1 Background and Notation

Let us start with the spherical K-Means optimization problem, which makes use of cosine similarity between datapoint and centroids to do the clustering. In this algorithm [37], each document, as well as each cluster centroid, is represented as a high-dimensional unit-length vector. Let $X = X_1, \ldots X_N$ be the datapoints, $C_1, \ldots C_K$ be the clusters, y_{ij} be an indicator variable specifying whether datapoint x_j belongs to cluster C_i, and μ_i denotes the centroid for cluster C_i. The task is to find optimal values of label assignments y_{ij} and cluster centroids μ_i, so as to optimize the objective function given in Equation 5.1.

$$\max_{y_{ij},\mu_i} \left(\sum_{j=1}^{N} \sum_{i=1}^{K} y_{ij} * cos(x_j, \mu_i) \right) \quad \text{s.t.} \sum_{i=1}^{K} y_{ij} = 1, \forall j = 1 \ldots N, y_{ij} \in \{0, 1\} \tag{5.1}$$

Since the variables y_{ij} and μ_i are dependent on each other, the above function can be optimized using an iterative EM like algorithm such that the E step finds the best of values of y_{ij} given μ_i's and the M step takes these y_{ij}'s to recompute the optimal μ_i's. Instead of probabilistically assigning a datapoint to a number of clusters, we choose hard assignments [28] i.e., we restrict y_{ij} as binary integers, hence this is the hard-EM algorithm. Fixing the values of μ_i's, the objective function is maximized when $y_{ij} = 1$ for $i = \textit{argmax}_i\ cos(x_j, \mu_i)$ and 0 otherwise. Let us now compute the μ_i that maximizes the objective given the values for y_{ij}'s. ($\langle a, b \rangle$ denotes dot product of vectors a and b.)

$$\max_{\mu_i} \left(\sum_{j=1}^{N} y_{ij} * cos(x_j, \mu_i) \right) = \max_{\mu_i} \left\langle \left(\sum_{j=1}^{N} y_{ij} \frac{x_j}{||x_j||} \right), \frac{\mu_i}{||\mu_i||} \right\rangle \tag{5.2}$$

The Cauchy-Schwartz inequality [48] states that $|\langle x, y \rangle| \leq ||x|| * ||y||$, and the two sides are equal only if x and y are linearly dependent. To maximize Equation 5.2, we can set $\mu_i = \left(\sum_j y_{ij} \frac{x_j}{||x_j||} \right)$. If we normalize x_j and μ_i to be unit vectors, then the equations can be simplified further. Given μ_i's, $y_{ij} = 1$ for $i = \textit{argmax}_i \left\langle x_j, \mu_i \right\rangle$ and 0 otherwise. Similarly, given y_{ij}'s, $\mu_i = \sum_j y_{ij} * x_j$. Henceforth, we assume that x_j and μ_i are normalized to be unit vectors. Hence, $cos(x_j, \mu_i) = \left\langle x_j, \mu_i \right\rangle$. Finally, the spherical K-Means algorithm repeats these two steps iteratively till convergence.

5.2.2 Multi-View K-Means

We extend the spherical K-Means algorithm [37] for the multi-view setting and present a general optimization based framework for Multi-View K-Means. Later, we also describe various ways of combining evidence from the different views to instantiate multiple variants of this general algorithm.

In the multi-view scenario, a datapoint x_j has a feature representation in each view denoted by $x_j^{(1)}$ in view 1 and $x_j^{(2)}$ in view 2. Once the datapoints are clustered in k clusters using K-Means representation, each centroid can also be represented differently in each view; i.e., centroid μ_i can be represented as $\mu_i^{(1)}$ in view 1 and $\mu_i^{(2)}$ in view 2. There is one score for every datapoint in every view; i.e., we will have scores $s_{ij}^{(1)}$ and $s_{ij}^{(2)}$ for datapoint x_j and centroid μ_i in view 1 and view 2 respectively to be defined below. There is a single label assignment vector per datapoint that combines scores from different views, which we represent as matrix Y, i.e., $y_{ij} = 1$ indicates that datapoint x_j belongs to cluster μ_i.

Let us define an *effective score* s_{ij} of datapoint x_j belonging to cluster μ_i as an overall measure of similarity of x_j to μ_i under both views. Then the optimization problem described in Equation 5.1 can be re-written as:

$$\max_{y_{ij}} \left(\sum_{j=1}^{N} \sum_{i=1}^{K} y_{ij} * s_{ij} \right) \quad \text{s.t.} \sum_{i=1}^{K} y_{ij} = 1, \forall j = 1 \ldots N, y_{ij} \in \{0, 1\} \tag{5.3}$$

In the multi-view scenario, the score s_{ij} can be defined as a function of $s_{ij}^{(1)}$ and $s_{ij}^{(2)}$. This can be done in various ways: e.g., a linear combination of scores, multiplication of scores, taking max or min over scores and so on.

Algorithm 9 describes a generic semi-supervised multi-view K-Means algorithm. It is different from standard K-Means in that each datapoint X_i and cluster centroid μ_j has a representation in each data view. In the E step of each EM iteration, for each unlabeled datapoint x we have a separate score from each view, a score of x belonging to a centroid μ_j. Line 8 in Algorithm 9 combines these scores and decides best cluster assignment for each datapoint. In the M step, centroids are recomputed per view based on the label assignments done in the E step. For simplicity of understanding, the algorithm is presented for two data views. However our techniques are easily applicable to datasets with more than two views.

Next, we will see various ways in which the cluster assignment step (line 8) and the centroid computation step (line 10) in this algorithm can be done, leading to new multi-view SSL algorithms. Recall that instead of probabilistically assigning a datapoint to a number of clusters, we choose hard assignments, i.e., we restrict $Y_t^u(x)$ (in Line 8) as binary integers, hence this is the hard-EM algorithm (or classification EM) [28]. Even though finding an optimal set of hard assignments using mixed integer programs is relatively more expensive than soft assignments, we are solving it per datapoint, instead of collectively optimizing labels for all datapoints. Hence we can divide the datapoints into multiple shards and parallelize the computation in Line 8 of Algorithm 9. Since we are using the spherical K-Means algorithm in a semi-supervised setting, henceforth we will use "cluster" and class" interchangeably.

Algorithm 9 Semi-supervised Multi-View K-Means Algorithm.

1: **function Multi-view-K-Means** ($X^{l(1)}$, $X^{l(2)}$, Y^l, $X^{u(1)}$, $X^{u(2)}$, k): $\mu^{(1)}$, $\mu^{(2)}$, Y^u
2: **Input**: $X^{l(1)}$, $X^{l(2)}$ labeled data points in view 1 and view 2 resp. with L_2 norm = 1; Y^l labels(or cluster assignments) for datapoints X^l; $X^{u(1)}$, $X^{u(2)}$ unlabeled datapoints in view 1 and view 2 resp. with L_2 norm = 1 (same feature space as X^l); k number of clusters to which the datapoints belong.
3: **Output:** $\mu^{(1)}$, $\mu^{(2)}$ cluster centroids in view 1 and view 2 resp. with L_2 norm = 1, $\mu^{(1)} = \{\mu_1^{(1)} \dots \mu_k^{(1)}\}$; Y^u labels(or cluster assignments) for unlabeled data points X^u
{Initialize model parameters using labeled data}
4: $t = 0$, $\mu_0^{(1)} = \textit{argmax}_\mu\ L(X^{l(1)}, Y^l|\mu)$,
$\mu_0^{(2)} = \textit{argmax}_\mu\ L(X^{l(2)}, Y^l|\mu)$, here L refers to the likelihood.
{here, $\mu_0^{(1)}$, $\mu_0^{(2)}$ are cluster centroids at iteration 0 in view 1,2 resp.}
5: **while** cluster assignments not converged, $t = t + 1$ **do**
{E step: (Iteration t) Make cluster predictions for the unlabeled data-points}
6: **for** $x \in X^u$ **do**
7: $s_j^{(1)} = \langle x^{(1)}, \mu_{j,t-1}^{(1)} \rangle$,
$s_j^{(2)} = \langle x^{(1)}, \mu_{j,t-1}^{(2)} \rangle$, for all labels $1 \le j \le k$
8: Compute cluster assignments $Y_t^u(x)$ given scores $s_{1\dots k}^{(1)}$,$s_{1\dots k}^{(2)}$.
9: **end for**
{M step : Recompute model parameters using current assignments for X^u}
10: Compute cluster centroids $\mu_t^{(1)}$, $\mu_t^{(2)}$ given $Y_t^u(x)$.
11: **end while**
12: **end function**

5.2.3 Cluster Assignment using Scores from Two Data Views

In this section, we go through various ways in which the cluster assignment and centroid computations steps (lines 8, 10) in Algorithm 9 can be done. We will start with an assumption that each datapoint belongs to exactly one cluster. (This can be viewed as a constraint that says "all classes $C_1 \dots C_k$ are mutually exclusive".) We also assume that the cluster assignment for a datapoint remains same in all data views. Later we will relax these assumptions one by one and present a way to solve these problem variants. Note that alternatives to these score combination methods are trivial methods like picking the best of available views, or concatenating feature vectors from different views. Later in Section 5.3 we will compare the performance of our proposed techniques against these baselines, and show the effectiveness of a carefully chosen score combination strategy.

SUMSCORE Method: Here we define the effective score as an addition of scores in different views: $s_{ij} = s_{ij}^{(1)} + s_{ij}^{(2)}$. Let us say $s_{ij}^{(1)} = \langle x_j^{(1)}, \mu_i^{(1)} \rangle$ and $s_{ij}^{(2)} = \langle x_j^{(2)}, \mu_i^{(2)} \rangle$.

$$\max_{\substack{y_{ij},\\ \mu_i^{(1)},\mu_i^{(2)}}} \left(\sum_{j=1}^{N} \sum_{i=1}^{K} y_{ij} * \left(\langle x_j^{(1)}, \mu_i^{(1)} \rangle + \langle x_j^{(2)}, \mu_i^{(2)} \rangle \right) \right) \text{ s.t. } \sum_{i=1}^{K} y_{ij} = 1 \ \forall j,\ y_{ij} \in \{0,1\}\ \forall i,j$$

In this setting the optimization objective can be written as follows:

$$\max_{\substack{y_{ij},\\ \mu_i^{(1)},\mu_i^{(2)}}} \left(\left\langle \sum_{j=1}^{N} y_{ij} * x_j^{(1)}, \mu_i^{(1)} \right\rangle + \left\langle \sum_{j=1}^{N} y_{ij} * x_j^{(2)}, \mu_i^{(2)} \right\rangle \right) \tag{5.4}$$

For the E step, keeping μ_i's fixed, the closed form solution to compute the y_{ij}'s is as follows: $y_{ij} = 1$ for $i = \textit{argmax}_i \left(\langle x_j^{(1)}, \mu_i^{(1)} \rangle + \langle x_j^{(2)}, \mu_i^{(2)} \rangle \right)$ and 0 otherwise. In the M step, once y_{ij}'s are fixed, the optimal μ_i's are those for which this objective function is maximized i.e., the two terms being summed are individually maximized. Each term is identical to Equation 5.2, and hence will have same solution. Given y_{ij}, we can compute $\mu_i^{(1)}$ and $\mu_i^{(2)}$ as follows: $\mu_i^{(1)} = \sum_j y_{ij} * x_j^{(1)}$, and $\mu_i^{(2)} = \sum_j y_{ij} * x_j^{(2)}$. Henceforth, we will refer to this method as SUMSCORE method.

PRODSCORE Method: Here we define the effective score as a product of scores in different views: $s_{ij} = s_{ij}^{(1)} * s_{ij}^{(2)}$. Keeping μ_i's fixed, we can compute y_{ij}'s as follows: $y_{ij} = 1$ for $i = \textit{argmax}_i \left(\langle x_j^{(1)}, \mu_i^{(1)} \rangle * \langle x_j^{(2)}, \mu_i^{(2)} \rangle \right)$ and 0 otherwise. Given y_{ij}, $\mu_i^{(1)}$ and $\mu_i^{(2)}$ can be computed as follows:

$$\max_{\mu_i^{(1)},\mu_i^{(2)}} \left(\sum_{j=1}^{N} y_{ij} * \left(\left\langle x_j^{(1)}, \mu_i^{(1)} \right\rangle * \left\langle x_j^{(2)}, \mu_i^{(2)} \right\rangle \right) \right) \tag{5.5}$$

Note that in Equation 5.5, unlike Equation 5.4, $\mu_i^{(1)}$ and $\mu_i^{(2)}$ cannot be independently optimized. However we can still follow an iterative procedure, that in the E step assigns $y_{ij} = 1$ for $i = \textit{argmax}_i \left(\langle x_j^{(1)}, \mu_i^{(1)} \rangle * \langle x_j^{(2)}, \mu_i^{(2)} \rangle \right)$ and 0 otherwise; and in the M step, recompute the centroids in different views as $\mu_i^{(1)} = \sum_j y_{ij} * x_j^{(1)}$, and $\mu_i^{(2)} = \sum_j y_{ij} * x_j^{(2)}$. Henceforth, we will refer to this method as the PRODSCORE method.

Next we propose yet another variant of the multi-view K-Means algorithm, which differs from the PRODSCORE and SUMSCORE methods in terms of their label assignment strategy in the E step of the EM algorithm.

MAXAGREE Method: The SUMSCORE and PRODSCORE methods assign the same label vector for each datapoint in both views and selects only one of the available labels (a hard mutual exclusion constraint). However for some datapoints, view 1 and view 2 might not agree on labels in the initial EM iterations; here we relax this constraint. Further, the mutual exclusion constraints between labels are also softened. Equation 5.6 (Figure 5.2 (a)) shows a new objective function that is being optimized for the entire dataset. Here, $f(y_{ij}^{(1)}, y_{ij}^{(2)}, s_{ij}^{(1)}, s_{ij}^{(2)})$ is a function of label vectors, and score vectors in the two views. Note that in this case multiple bits in y_i can be 1, and finding the best possible bit vector of y_i's can lead to evaluating 2^k possible assignments.

Equation 5.7 (Figure 5.2 (b)) shows an instantiation of Equation 5.6 and gives the mixed integer linear program (MIP) that is solved per datapoint. This new method, called MAXAGREE (maximize agreement) allows different views to assign a datapoint to different clusters, with a penalty on cluster assignments being different. In particular, label predictions are done by solving a MIP on scores produced in the two views and choosing a label vector per datapoint per data view, with a penalty for assigning different labels across views and a penalty for not following

(a)
$$\underset{y_{ij}^{(1)},y_{ij}^{(2)},\mu_i^{(1)},\mu_i^{(2)}}{\text{maximize}} \left(\sum_j \left(f(y_{ij}^{(1)}, y_{ij}^{(2)}, s_{ij}^{(1)}, s_{ij}^{(2)}) - \left(\text{Penalty for}\left(\sum_{i=1}^{K} y_{ij}^{(1)} \neq y_{ij}^{(2)}\right), \left(\sum_{v=1}^{2}\sum_{i=1}^{K} y_{ij}^{(v)} \neq 1\right)\right)\right)\right)$$
$$\text{subject to,} \quad y_{ij}^{(1)} \in \{0,1\}, y_{ij}^{(2)} \in \{0,1\} \tag{5.6}$$

(b)
$$\underset{y_i^{(1)},y_i^{(2)},d_i,\zeta^1,\zeta^2,\delta^1,\delta^2}{\text{maximize}} \left(\alpha_1 * \left(\sum_{\mathrm{i}} \mathrm{y_i^{(1)}} * \mathrm{s_i^{(1)}} + \mathrm{y_i^{(2)}} * \mathrm{s_i^{(2)}}\right) - \alpha_2 * \left(\sum_{\mathrm{i}} \mathrm{d_i}\right) - \alpha_3 * \left(\zeta^1 + \zeta^2 + \delta^1 + \delta^2\right)\right)$$
$$\text{subject to, } d_i = |y_i^{(1)} - y_i^{(2)}|, \forall i = 1 \ldots k$$
$$\sum_i y_i^{(1)} \leq 1 + \delta^1, \quad \sum_i y_i^{(2)} \leq 1 + \delta^2, \quad \forall i = 1 \ldots k$$
$$\sum_i y_i^{(1)} \geq 1 - \zeta^1, \quad \sum_i y_i^{(2)} \geq 1 - \zeta^2, \quad \forall i = 1 \ldots k$$
$$\zeta^1, \zeta^2, \delta^1, \delta^2 \geq 0, \quad y_i^{(1)} \in \{0,1\}, \quad y_i^{(2)} \in \{0,1\} \quad \forall i \tag{5.7}$$

Figure 5.2: (a) Optimization formulation for multi-view learning (b) Mixed integer program for MAXAGREE method with two views.

the mutual exclusion constraints. Further, Equation 5.6 can be independently optimized for each datapoint.

For each datapoint $s_i^{(1)}$ and $s_i^{(2)}$ are scores of x belonging to cluster i according to data view 1 and view 2 respectively. $y_i^{(1)}$ and $y_i^{(2)}$ represent cluster assignments in view 1 and view 2 respectively. The term d_i is the penalty on $y_i^{(1)}$ and $y_i^{(2)}$ being different. Terms ζ^1 and ζ^2 are the penalty terms for the constraint that each datapoint should be assigned to at least one cluster in each view. Similarly, δ^1 and δ^2 are the penalty terms for the constraint that each datapoint should be assigned to at most one cluster in each view. Terms α_1, α_2 and α_3 are constants that define relative importance of terms in the objective function.

COTRAIN Method: We also experiment with a well-studied co-training based method CO-TRAIN as one of the baselines. This method is a multi-view spherical K-Means algorithm that is proposed by Bickel and Scheffer [10]. In particular, it is a co-training variant of the spherical K-Means algorithm, in which predictions made for view 1 in the E step are used to recompute centroids of view 2 in the M step and visa versa.

5.3 Datasets and Experimental Methodology

Here, we first describe the datasets used. We then explain the experimental setup and evaluation criteria used.

5.3.1 Datasets

We present experiments with eight multi-view datasets, summarized in Table 5.1. The first six of them are publicly available, and the last two were created by us for exploring the multi-view learning task. Cora, WebKB and CiteSeer$_1$ [52] consists of two data views for scientific publications. The first view consists of 0/1-valued word vector derived from the document text, and the other view is citation links between these documents (represented as binary features).

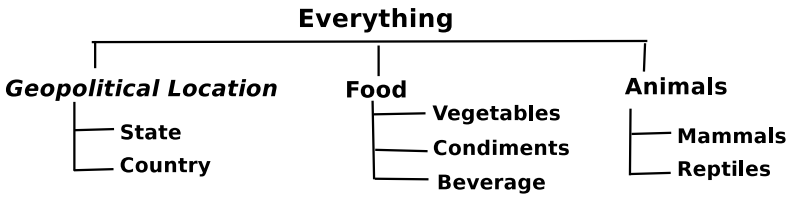


Figure 5.3: Class ontology used in the NELL_mv dataset.

Each document in the Cora, WebKB and CiteSeer$_1$ datasets has been classified into one of the seven, five and six classes respectively. The next two datasets, Wikipedia_linqs and Pubmed_linqs [52] also contain a bag of words and the links between documents as the two views, but they are different from the earlier datasets in the sense that words features are TF/IDF-weighted instead of being 0/1-valued. Wikipedia_linqs has 19 distinct categories, whereas the Pubmed_linqs dataset has 3 categories.

The UCI Leaves image dataset [81] contains sixteen samples of leaves of each of one-hundred plant species. For each sample, a shape descriptor and texture histogram (feature vectors of size 64 each) are given. The Citeseer$_2$ dataset contains text and author views for around 6K scientific articles, classified into 17 categories. The NELL_mv dataset was created using around 1800 entities in the NELL KB, and the data views are occurrences of those entities in ClueWeb09 text and HTML table data [23, 126]. The NELL_mv dataset contains hierarchical labels that can be arranged in an ontology shown in Figure 5.3. For the experiments in this chapter, we use labels at a particular level of hierarchy, while for hierarchical multi-view learning experiments (see Chapter 6.2.2) we make use of the ontology to create class constraints.

General statistics like the number of datapoints, features, classes etc. about these datasets are summarized in Table 5.1. We can see that our datasets have varying number of datapoints and classes. They also cover different kinds of features like: binary text features, tfidf text features, link features, semi-structured data features and image features. We have made the hierarchical multi-view NELL_mv dataset available[11] for the research community to help future research in this field.

5.3.2 Experimental Setting

In addition to the SUMSCORE and PRODSCORE baselines, we experimented with three other single view baselines. Methods V1 and V2 are single-view spherical K-Means methods that use only view 1 and view 2 of the data respectively. For the experiments in this section we order the views for each dataset such that view 1 is on average better than view 2 in terms of F1 scores. Using the concatenation of two views as feature vectors yields Method V12. Method COTRAIN is the co-training based multi-view spherical K-Means algorithm proposed by Bickel and Scheffer [10]. We compared these baseline methods with multi-view methods based on our proposed optimization formulation: These methods include SUMSCORE, PRODSCORE, and MAXAGREE (Section 5.2.3).

[11]The dataset can be downloaded from `http://rtw.ml.cmu.edu/wk/WebSets/multiView_2015_online/index.html`

Dataset	#Datapoints	#Classes	View statistics			
			View 1		View 2	
			#Features	#(Datapoint, feature) pairs	#Features	#(Datapoint, feature) pairs
Cora	2708	7	1433	49.2K	2222	5.4K
WebKB	877	5	1703	79.4K	735	1.6K
Citeseer$_1$	3312	6	3703	105.1K	2312	4.7K
Wikipedia_linqs	3363	19	4973	2.1M	2851	45.0K
Pubmed_linqs	19.7K	3	500	988.0K	19.7K	44.3K
UCI Leaves	1600	100	64	102.0K	64	102.4K
Citeseer$_2$	6601	17	26.7K	392.0K	10.8K	16.0K
NELL_mv	1855	11	3.4M	8.8M	1.1M	2.4M

Table 5.1: Statistics of all datasets used.

5.3.3 Evaluation Criteria

To evaluate the performance of various methods, we use macro and micro averaged F1 scores. The *macro averaged F1 score* gives equal weight to performance on all classes, hence is better in cases where class distribution is skewed. On the other hand, the *micro averaged F1 score* gives equal weight to performance on all datapoints, hence it is biased towards the frequent classes. For each dataset we experimented with different values of the training percentage. For each value of training percentage, we average the scores across 10 random train/test partitions. Note that we are running these experiments in the transductive setting i.e., our semi-supervised learning algorithm will get as input training examples with ideal labels as well as unlabeled test examples; and the task is to learn a model from labeled training and unlabeled test data to in turn label the test datapoints.

For the multi-view experiments presented in Section 5.4.1, we run experiments on all 8 datasets, with 2 values of training percentages $\{10, 30\}$. For the NELL_mv dataset we do separate evaluation at the second and third levels of the class hierarchy. Hence for each method we get 18 values of average F1 scores for each of the macro and micro averaging methods.

Next, we compute the *average rank* of methods for these 18 test-cases. While computing average rank, we first compute a method's average rank according to macro-averaged F1 scores, and micro-averaged F1 scores, and take the average of these 2 ranks. The lower the average rank, the better the method's performance. Further we visualize these results using scatter plots that show the F1 score of each of the proposed method vs. baseline method. The proposed methods are shown on the x axis compared to baselines on the y axis, hence points below the 'x=y' dotted line mean that the proposed methods performed better. Both average rank and scatter plots help us measure the effectiveness of methods over a number of datasets.

Finally, we also present correlation analysis of performance improvements produced by our techniques w.r.t dataset characteristics like View agreement, and View imbalance. view agreement is defined as the fraction of datapoints for which methods V1 and V2 produce same labels; its value varies between 0 to 1. We define view imbalance as the difference between the performances of the two views; its value can vary between 0 to 100. Consider an example where on a particular dataset, V1 and V2 gave F1 score of 55.0 and 42.7 respectively, then view imbalance

is $|55.0 - 42.7| = 12.3$. If the final labels produced by V1 and V2 agree on 500 out of 1000 datapoints, then the view agreement is $500/1000 = 0.5$.

5.4 Experimental Results

In this section we go over our experimental findings.

5.4.1 Comparison of Multi-view Learning Methods in Terms of F1

Table 5.2 compares the macro-averaged F1 scores of the proposed optimization based methods with baseline methods on all 8 multi-view datasets with 10% and 30% training data. V12 method came out as a strong baseline. We can see that our proposed methods SUMSCORE, and MAXAGREE give comparable or better performance w.r.t all baseline methods, and clearly outperform the COTRAIN and V1 (best of two views) methods on most datasets. We can also see that the MAXAGREE method performs as well as and sometimes better than the SUMSCORE method. MAXAGREE allows different label vectors to be assigned in different views and hence is the most expressive of the proposed methods.

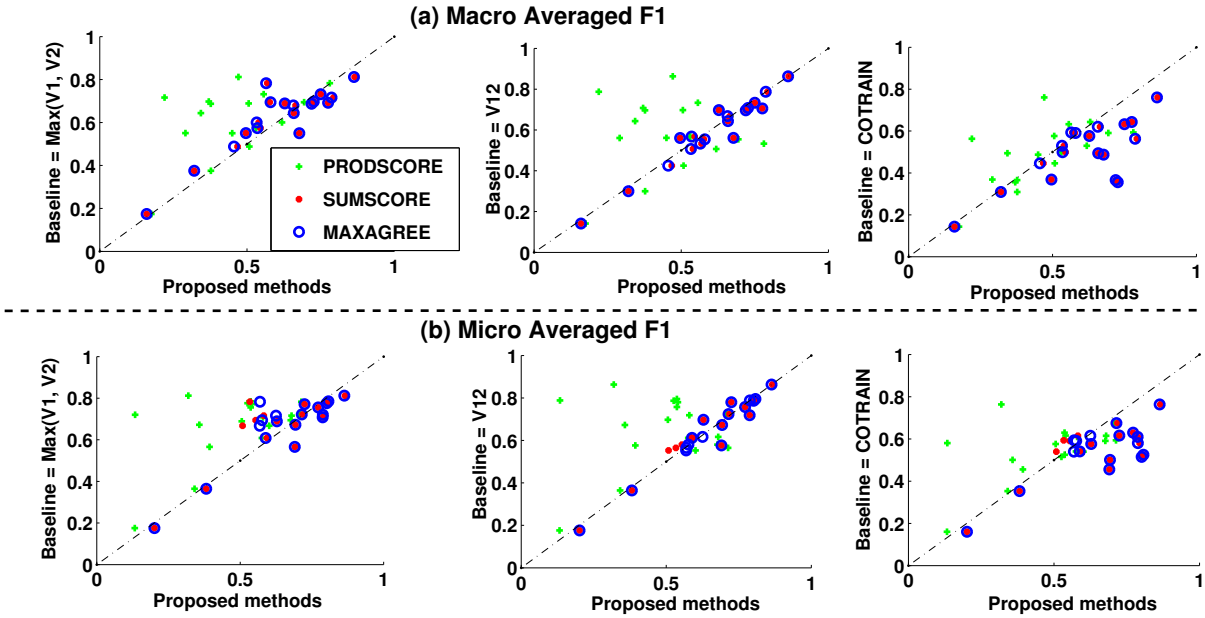


Figure 5.4: Scatter plots of F1 scores of Flat Multi-view methods on all datasets. Proposed methods (PRODSCORE, SUMSCORE, MAXAGREE) are on the x axis compared to baselines (Max(V1,V2), V12, COTRAIN) on the y axis, hence points below the 'x=y' dotted line mean that the proposed method performed better.

Figure 5.4 shows the scatter plot of the macro and micro averaged F1 scores of our proposed methods (PRODSCORE, SUMSCORE, and MAXAGREE) vs. baselines (V1, V2, V12, and COTRAIN). Points below x=y line denotes that our proposed method (plotted on the x axis) performed better than the baseline method (plotted on the y axis). These plots reinforce the fact

Dataset	Train percentage	View Agreement	Baseline methods V1	V2	V12	CO-TRAIN	Proposed methods PROD-SCORE	SUM-SCORE	MAX-AGREE
Cora	10	0.30	55.0	42.7	56.2	48.8	45.0	**67.8+**	**67.8+**
	30	0.50	69.2	62.1	70.6	64.3	63.1	**77.6+**	**77.6+**
WebKB	10	0.32	55.1	29.0	**56.1**	36.8	29.1	49.6	49.6
	30	0.50	68.9	50.0	**69.8**	57.6	50.6	62.8	62.8
Citeseer_1	10	0.35	64.4	33.3	64.4	49.4	34.3	**65.9+**	**65.9+**
	30	0.52	73.2	54.9	73.3	63.3	55.7	**75.0+**	**75.0+**
Wikipedia_linqs	10	0.44	**57.4**	42.0	56.8	49.9	53.1	54.6	53.5
	30	0.60	**68.0**	55.7	66.8	62.0	66.6	66.5	65.7
Pubmed_linqs	10	0.67	69.8	37.0	70.8	35.6	37.0	**72.7+**	**72.7+**
	30	0.77	68.7	37.7	69.6	36.6	37.7	**71.9+**	**71.9+**
UCI Leaves	10	0.42	71.6	56.8	78.7	56.3	22.0	**79.4+**	78.7+
	30	0.59	81.2	70.6	86.3	76.1	47.1	**86.6+**	86.3+
Citeseer_2	10	0.13	17.4	14.1	14.1	14.4	**17.5+**	16.0+	16.0+
	30	0.33	37.5	30.0	30.0	30.9	**37.7+**	32.1	32.1
NELL_mv level=2	10	0.66	**78.3**	53.2	53.3	59.3	78.1	56.9	56.5
	30	0.79	**69.4**	55.1	55.4	59.1	**69.4+**	57.7	58.0
NELL_mv level=3	10	0.45	48.8	42.7	42.5	44.6	**50.8+**	46.8	45.6
	30	0.57	60.0	50.2	50.7	53.0	**61.9+**	54.0	53.3
Avg. Rank			3.6	5.8	**3.3**	5.1	4.6	**2.7**	**2.9**

Table 5.2: Comparison of proposed optimization based multi-view methods w.r.t baselines on all 8 datasets in terms of macro-averaged F1 scores. Best F1 scores in each row are bold-faced. (+) in front of scores for proposed methods indicate that for that dataset the proposed method performed better than or equal to the best baseline score for the dataset. Last row of the table shows average rank of each method in terms of both macro averaged F1 and micro averaged F1 scores. Top 3 method ranks are bold-faced.

that the SUMSCORE and MAXAGREE methods are performing comparable to or better than all baseline methods whereas the PRODSCORE method does not consistently outperform the baselines.

We also compare these methods in terms of their average rank according to both macro averaged and micro averaged F1 scores. According to this metric, the top 3 methods are SUMSCORE, MAXAGREE and V12, in that order.

Thus we observed that our optimization based label assignment strategies (SUMSCORE and MAXAGREE) give state-of-the-art or comparable performance when compared to existing flat multi-view techniques, including the well studied co-training based baseline COTRAIN. In terms of average rank of the baselines V12 performed best of the baselines.

5.4.2 Analysis of Results

From Table 5.2 and scatter plots in Figure 5.4, we can see that different methods gave the best performance on different datasets. To help understand the reasons for this, we studied the cor-

Correlation with	% relative improvement	correlation	pVal
View imbalance	PRODSCORE over V12	**-0.50**	**0.03**
	MAXAGREE over V12	-0.39	0.11
	MAXAGREE over PRODSCORE	0.36	0.14

Table 5.3: Flat Classification: Pearson Correlation coefficient between the view imbalance and performance improvements produced by multi-view methods over the baselines.

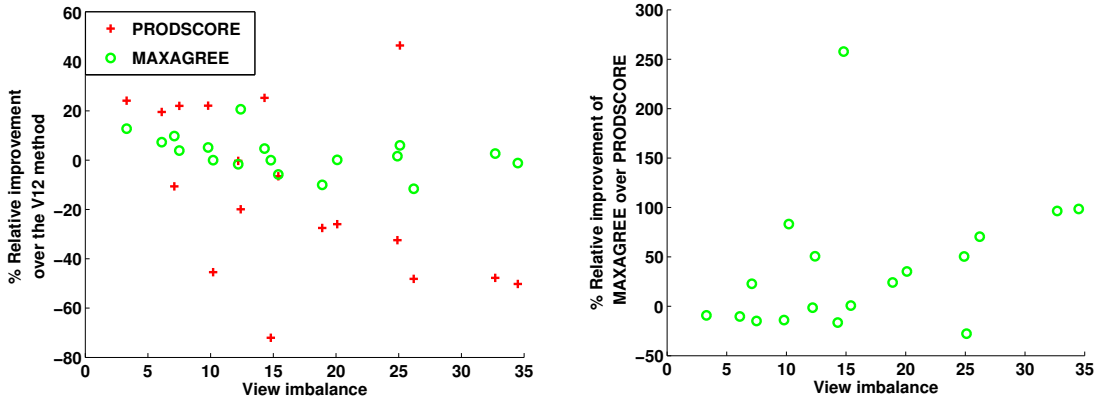


Figure 5.5: Percentage relative improvement of multi-view methods for datasets with varying view imbalance.

relation of performance improvements of proposed methods (over baseline V12) w.r.t the view imbalance. As described in Section 5.3.3, view imbalance is defined as the difference between the performances of V1 and V2. From Figure 5.5 and Table 5.3, we can see that the performance improvements of both PRODSCORE and MAXAGREE over V12 are negatively correlated with the difference between the two views, i.e., there is less improvement over the baseline when there is larger view imbalance (as the performance of two views differ by larger amount). This seems natural as simpler score combination methods (like best view or concatenation of views) should work well when most information about an example is in only one view. Further we can see that the improvement of MAXAGREE over PRODSCORE is higher with larger view imbalance; we found a weak positive correlation of 0.36 with pVal = 0.14.

Prior work [33] has also studied correlation between the performance of multi-view methods and view agreement. The third column of Table 5.2 gives the value of agreement between methods V1 and V2. We could not find any significant correlation between the view agreement and the improvements of proposed methods w.r.t. baselines. This might be due to the fact that the datapoints for this correlation analysis are coming from different kinds of datasets. When we did similar analysis in the subsequent hierarchical experiments on the single NELL_mv dataset, we found a significant correlation between agreement rate and performance improvement of proposed methods w.r.t. baselines (refer to Figure 6.3).

5.5 Related Work

Multi-view learning as defined by Xu et al. [144] is a paradigm that introduces a different function to model each view and jointly optimizes all the functions to exploit the redundant views of

the same input data and improves the learning performance.

Blum and Mitchell [13] proposed the co-training algorithm for problems where the examples are described by two conditionally independent views. It jointly trains two classifiers such that classifier A adds examples to the labeled set that classifier B will then be able to use for learning. If the two views are conditionally independent, then co-training will always improve the results, otherwise it may not be successful. Later, Nigam and Ghani [97] analyzed the performance of co-training when certain assumptions are violated. More generally, one can define a learning paradigm that utilizes the agreement among different learners, and the particular assumptions of co-training are not required. Bickel and Scheffer proposed a multi-view spherical K-Means algorithm [10]. In particular, it is a co-training variant of the spherical K-Means algorithm, in which predictions made for view 1 in the E step are used to recompute centroids of view 2 in the M step and visa versa. We used this method as a baseline (referred to as COTRAIN in Section 5.3).

Instead, multiple hypotheses with different inductive biases, e.g., decision trees, SVMs, etc. can be trained from the same labeled data set, and are required to make similar predictions on any given unlabeled instance. Sindhwani et al. [114] and Brefeld et al. [17] proposed multi-view semi-supervised regression techniques.

Another related area of research is multi-task learning. Jin et al. [66] proposed a single framework to incorporate multiple views and multiple tasks by learning shared predictive structures. On similar lines, Hu et al. [62] handle the problem of heterogeneous feature spaces in the context of transfer learning, and show improved performance on the tag recommendation task. Kang and Choi [69] developed a method based on restricted deep belief networks for multi-view problems, such that layer of hidden nodes in the belief network have view-specific shared hidden nodes. Usunier et al. [131] focus on learning to rank multilingual documents, using machine translation of documents in other languages as different data views. Several boosting approaches like Mumbo [72] and ShareBoost [103] are also proposed for multi-view learning problems. Our techniques are weakly supervised and do not assume the amount of training data required to train such boosting algorithms.

Recent research on multiple kernel learning has proposed a number of approaches for combining kernels in the regularized risk minimization framework [38, 45, 78]. Researchers have also explored dimensionality reduction techniques to create unified low-dimensional embeddings from multi-view datasets so as to benefit semi-supervised learning and information extraction tasks [122, 125]. Cai et al. [22] proposed the use of structured sparsity inducing norms to make K-Means algorithm run on large datasets using multi-threaded machines. Their method is complementary to our techniques because we focus on consistent label assignment for a single datapoint in the E step of every K-Means iteration, which can be incorporated in their multi-threaded setting.

Brefeld et al. [15, 16] proposed multi-view learning techniques for more challenging structured output spaces. Our methods are different in that we make use of the well-studied EM framework, pose the label assignment in the E step as an optimization problem, and propose multiple linear and mixed-integer formulations to solve such optimization problem. Gilpin et al. [54] have proposed a integer linear programming based method for hierarchical clustering. Our techniques are different in that Gilpin et al. focus on unsupervised agglomerative clustering whereas we focus on semi-supervised and multi-view clustering in the presence of predefined

class hierarchy.

5.6 Conclusions

In this chapter, we investigated the problem of semi-supervised learning in the presence of multiple-data views. We formulated the problem as an optimization problem, and solved it using the standard EM framework. We then focused on the sub-problem of assigning labels to each datapoint (part of E step), and studied various methods for such prediction. Our proposed method solves a mixed integer linear program to find consistent class assignments given the scores in each data view. Because our multi-view techniques are broadly similar to co-training based algorithms, we also compared them to a seeded version of multi-view spherical K-Means algorithm that is proposed by Bickel and Scheffer [10]. Our methods produced better performance in terms of macro-averaged F1 score compared to this co-training baseline.

We present extensive experiments on 8 different multi-view datasets, including document classification, image classification, publication categorization and knowledge based information extraction datasets. The data views also vary in their nature including word occurrences, binary features, link features, image histograms and co-occurrence features. We showed that our techniques give state-of-the-art performance when compared to existing multi-view learning methods including the co-training based algorithm proposed by Bickel and Scheffer [10], on the problem of flat multi-view semi-supervised learning.

Finally, we have made the hierarchical multi-view NELL_mv dataset available[12]. We have published the text context and HTML table features, class hierarchy, hierarchical labels for all entities, and seed train-test partitions of NELL_mv dataset used in this thesis.

[12]Dataset link: `http://rtw.ml.cmu.edu/wk/WebSets/multiView_2015_online`

Chapter 6

Constrained Semi-supervised Learning in the Presence of Ontological Constraints

In the last chapter, we discussed semi-supervised learning techniques that can make use of the multi-view datasets. We also found that such multi-view models are more effective than models learnt using any single view or concatenation of views. However, multiple views are only one issue arising in complex real-world learning tasks. For instance, in the earlier mentioned KB population task, the labels assigned to each noun-phrase need to be consistent with the hierarchical class constraints posed by the KB ontology.

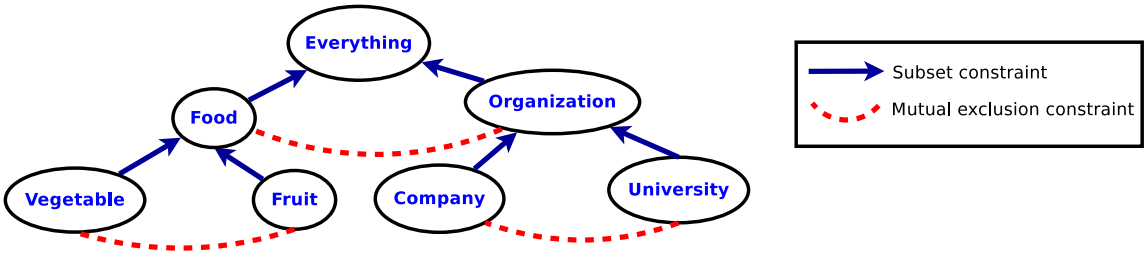


Figure 6.1: An example of ontological with subset and mutual exclusion relations between classes.

Consider a toy example of ontological class constraints in Figure 6.1. Here, we can see two kinds of class constraints imposed by the ontology. (1) The "Subset" constraint between "Fruit" and "Food" categories suggests that if a datapoint is classified as "Fruit", then it should also be classified as "Food". (2) The "Mutual Exclusion" constraint between "Food" and "Organization" says if a datapoint is classified as "Food", then it should not be classified as "Organization", and vice versa. Further, the combination of these constraints induce many more constraints. For example, "Food" is mutually exclusive with "Organization" and "Company" is subset of "Organization", implies that "Vegetable" and "Company" classes are mutually exclusive.

Thus, while classifying the noun-phrase "Carnegie Mellon University" w.r.t. class ontology in Figure 6.1, we need to choose the labels Everything, Organization and University (consistent with the class constraints) while combining clues from both data views (text-patterns and HTML-tables).

6.1 Hierarchical Semi-supervised Learning

Similar to the multi-view semi-supervised learning techniques discussed in Chapter 5, we are formulating the semi-supervised learning with ontological constraints as a constrained learning task. In this task, each datapoint is assigned a bit vector of labels, one bit per class in the ontology. The ontological constraints tell us which bit vectors are consistent and which are invalid. Further, our proposed optimization method is not limited to tree-structured class hierarchies. It can also deal with non-tree class hierarchies defined by sets of subset and mutual exclusion constraints. Next, we discuss this procedure in detail (Section 6.1.1).

6.1.1 Method

We propose Hier-Semi-EM, a hierarchical semi-supervised Expectation Maximization (EM) algorithm. It uses small amount of labeled data and large amount of unlabeled data in an iterative EM framework to learn a hierarchical model for this task. The algorithm is presented in Algorithm 10.

Algorithm 10 Hierarchical Semi-supervised Learning.

function Hier-Semi-EM (X^l, Y^l, X^u, $\{C_1 \dots C_k\}$, Z_k): $\theta_1 \dots \theta_k$, Y^u
Input: X^l labeled data points; Y^l labels of X^l; X^u unlabeled data points;
$\{C_1 \dots C_k\}$ set of known classes to which x's belong;
Z_k manually input constraints on k seed classes;⟸
Output: $\theta_1 \dots \theta_k$ parameters for k classes; Y^u labels for X^u
{Initialize classifiers θ_j for class C_j using seeds provided for C_j}
$\theta_1^0 \dots \theta_k^0 = argmax_\theta L(X^l, Y^l)$
while Class assignments not converged **do**
{E step: (Iteration t) Classify each datapoint at each level of class hierarchy}
for $i = 1$ to $|X^u|$ **do**
Find $P(C_j|X_i, \theta_1^{(t)}, \dots, \theta_k^{(t)})$ for all labels $1 \leq j \leq k$
$Y_i^{(t)}$ = **ConsistentAssignment**($P(C_j|X_i)$, h, Z_k) ⟸
end for
{M step: Recompute model parameters based on $Y^{(t)}$ }
$\theta_k^{(t+1)} = argmax_\theta L(X^l, Y^l, X^u, Y^{u(t)}|\theta_k^{(t)})$
end while
end function

Before getting into the details of the algorithm, let us review the notations. Let $X = \{X_1, \ldots X_N\}$ be the datapoints to be classified, and $\{C_1, \ldots C_K\}$ be the KB categories. Let $y_{ji} \in \{0, 1\}$ be an indicator variable specifying whether a datapoint X_i belongs to category C_j. Let θ_j denote the centroid/classifier for category C_i. Using the model parameters θ_j for class C_j, we can estimate $P(C_j|X_i)$, the probability of X_i belonging to category C_j. Hier-Semi-EM aims to find optimal values of label assignments y_{ji} and model parameters θ_j, so as to maximize the overall data likelihood. Let *Subset* be the set of all subset or inclusion constraints, and *Mutex* be the set of all mutual exclusion constraints. In other words, *Subset* = $\{\langle i, k\rangle : C_i \subseteq C_k\}$ and *Mutex* = $\{(i, k) : C_i \cap C_k = \phi\}$.

Hier-Semi-EM takes in as input the set of datapoints X, out of which X^l is the labeled subset with category labels in Y_l, and the remaining X^u is unlabeled. Additionally, it takes as input a set of ontological constraints, Z. In the E step of this algorithm, each datapoint X_i is assigned a bit vector of labels $Y_i = [y_{1i}, \ldots y_{ki}]$. The class hierarchy is incorporated as constraints on bits in this bit vector. For example, if for a datapoint *cat*, a bit corresponding to KB category *mammal* is set then the bit corresponding to category *animal* should also be set (subset constraint: *mammal* $\subseteq$ *animal*), and for the same datapoint the bit corresponding to category *reptile* should not be set (mutual exclusion constraint: mammal $\cap$ reptile = ϕ). The M step recomputes the model parameters for each KB category using the label assignments done in the E step.

The E step of Hier-Semi-EM may be broken down into two stages, which we describe in more detail below:

- Algorithm 10 Line 7: This step computes probabilities $P(C_j|X_i; \theta_j)$, where θ_j is the current estimate of model parameters for category C_j. A variety of techniques may be used for this estimation. We briefly describe one such choice here: the semi-supervised version of multinomial Naive Bayes [98]. In this model $P(C_j|X_i) \propto P(X_i|C_j) * P(C_j)$, for each unlabeled gloss X_i. The probability $P(X_i|C_j)$ is estimated by treating each feature in X_i as an independent draw from a class-specific multinomial. In this task, the features are word occurrences in the candidate glosses, and the number of outcomes of the multinomial is the vocabulary size. The Naive Bayes version can be referred to as Hier-Semi-EM-NB, whereas the K-Means version is referred to as Hier-Semi-EM-KM.
- ConsistentAssignment (Refined Label Assignment using Ontological Constraints) (Algorithm 10 Line 8): Given the category memberships probabilities $\{P(C_j|X_i)\}$ estimated above, this step computes the category membership variables $\{y_{ji}, \forall\, 1 \leq i \leq N,\, 1 \leq j \leq K\}$. Hier-Semi-EM solves a Mixed-Integer Program (MIP) to estimate these variables. One such problem is solved for each datapoint. This MIP takes the scores $\{P(C_j|X_i)\}$, and class constraints Z as input and produces a bit vector of labels as output, each bit representing whether the datapoint belongs to that particular category.

  The MIP formulation for a datapoint X_i is presented in Equation 6.1. For each X_i, this method tries to maximize the sum of scores of selected labels, after penalizing for violation of class constraints. Let ζ_{jk} are slack variables for *Subset* constraints, and δ_{jk} are slack variables for *Mutex* constraints.

$$\underset{\{y_{ji}\},\zeta_{jk},\delta_{jk}}{\text{maximize}} \sum_{\text{j}} \text{y}_{\text{ji}} * \text{P}(\text{C}_\text{j}|\text{X}_\text{i}) - \sum_{\langle \text{i,k}\rangle \in \text{Subset}} \zeta_{\text{ik}} - \sum_{(\text{i,k}) \in \text{Mutex}} \delta_{\text{ik}}$$

$$\begin{aligned} \text{subject to,} \\ y_{ji} \geq y_{ki} - \zeta_{jk}, \quad \forall \langle j,k \rangle \in \textit{Subset} \\ y_{ji} + y_{ki} \leq 1 + \delta_{jk}, \quad \forall (j,k) \in \textit{Mutex} \\ \zeta_{jk}, \delta_{jk} \geq 0,\ y_i \in \{0,1\}, \quad \forall j,k \end{aligned} \tag{6.1}$$

6.1.2 Experimental Results: Do ontological constraints help?

Table 6.1 shows the comparison between semi-supervised EM algorithm with and without ontological constraints. The best values in each row (per dataset per level in the hierarchy) are bold-faced. We can see that for 9 out of 10 cases, the best performance was given by a method that uses ontological constraints while clustering. Hence we can say that ontological constraints do help in this learning task. The Hier-Semi-EM algorithm is on average 8.7 times more expensive than EM in terms of average runtime.

Dataset	Level	Macro-avg. seed class F1	
		EM	Hier-Semi-EM
Text-Small	2	46.6	**52.0**
	3	23.5	**25.8** △
Text-Medium	2	53.2	**53.3**
	3	27.9	**33.9** ▲
	4	17.4	**26.8** △
Table-Small	2	69.5	**74.8** ▲
	3	36.8	**38.9** ▲
Table-Medium	2	**62.7**	62.2
	3	43.7	**48.0** ▲
	4	47.3	**57.1** ▲

Table 6.1: Comparison of flat vs. hierarchical semi-supervised EM methods using KM representation on Text-Small to Table-Medium.

6.2 Multi-view Learning in the Presence of Ontological Constraints

In Chapter 5 we discussed semi-supervised learning techniques that can make use of the multi-view datasets. The proposed techniques make use of linear optimization based formulations to incorporate evidence from multiple data views. In this section, we present their hierarchical variants that use a class hierarchy in the learning process to improve over their flat counterparts. This shows the potential of such linear optimization based formulations for complex learning

scenarios that cover multi-view and hierarchical classification in a single framework. This work is currently in preparation for submission [123].

6.2.1 Hierarchical Multi-view Learning

In this section we will discuss natural extensions of our proposed multi-view approaches for hierarchical multi-view learning. The only change that needs to be made is that each datapoint can be assigned multiple labels, so that they satisfy the class constraints. Further, instead of making the constraints hard, we relax them using slack variables. These slack variables add a penalty to the objective function upon violating any of the class constraints. Let *Subset* be the set of all subset or inclusion constraints, and *Mutex* be the set of all mutual exclusion constraints. In other words, *Subset* = $\{\langle i,j\rangle : C_i \subseteq C_j\}$ and *Mutex* = $\{(i,j) : C_i \cap C_j = \phi\}$. Note that our only assumption is that we know the subset and mutual exclusion constraints between classes under consideration. However, we do not assume that the classes are necessarily arranged in a tree structured hierarchy. Next, we will discuss three methods that do multi-view learning in the presence of such class constraints.

(a)
$$\begin{aligned} &\underset{y_i,\zeta_{ij},\delta_{ij}}{\text{maximize}} \left(\sum_{\mathrm{i}} \mathrm{y_i} * (\mathrm{s_i^{(1)}} + \mathrm{s_i^{(2)}}) - \sum_{\langle \mathrm{i,j}\rangle \in \mathrm{Subset}} \zeta_{\mathrm{ij}} - \sum_{(\mathrm{i,j})\in \mathrm{Mutex}} \delta_{\mathrm{ij}} \right) \\ &\text{subject to, } y_j \geq y_i - \zeta_{ij}, \quad \forall \langle i,j\rangle \in \textit{Subset} \\ &y_i + y_j \leq 1 + \delta_{ij}, \, \forall (i,j) \in \textit{Mutex}, \quad \zeta_{ij}, \delta_{ij} \geq 0, \quad y_i \in \{0,1\} \, \forall i,j \end{aligned} \tag{6.2}$$

(b)
$$\begin{aligned} &\underset{y_i^{(1)},y_i^{(2)},d_i,\zeta_{ij}^{(1)},\delta_{ij}^{(1)},\zeta_{ij}^{(2)},\delta_{ij}^{(2)}}{\text{maximize}} \left(\alpha_1 * \left(\sum_{\mathrm{i}} \mathrm{y_i^{(1)}} * \mathrm{s_i^{(1)}} + \mathrm{y_i^{(2)}} * \mathrm{s_i^{(2)}} \right) - \alpha_2 * \sum_{\mathrm{i}} \mathrm{d_i} \right. \\ &\qquad \left. -\alpha_3 * \left(\sum_{\langle i,j\rangle \in \textit{Subset}} (\zeta_{ij}^{(1)} + \zeta_{ij}^{(2)}) + \sum_{(i,j)\in \textit{Mutex}} (\delta_{ij}^{(1)} + \delta_{ij}^{(2)}) \right) \right) \\ &\text{subject to, } d_i = |y_i^1 - y_i^2|, \quad \forall i = 1 \ldots k \\ &y_j^{(1)} \geq y_i^{(1)} - \zeta_{ij}^{(1)}, \quad y_j^{(2)} \geq y_i^{(2)} - \zeta_{ij}^{(2)}, \quad \forall \langle i,j\rangle \in \textit{Subset} \\ &y_i^{(1)} + y_j^{(1)} \leq 1 + \delta_{ij}^{(1)}, \quad y_i^{(2)} + y_j^{(2)} \leq 1 + \delta_{ij}^{(2)}, \quad \forall (i,j) \in \textit{Mutex} \\ &\zeta_{ij}^{(1)}, \delta_{ij}^{(1)}, \zeta_{ij}^{(2)}, \delta_{ij}^{(2)} \geq 0, y_i \in \{0,1\} \, \forall i,j \end{aligned} \tag{6.3}$$

Figure 6.2: Mixed integer program for (a) Hier-SUMSCORE method and (b) Hier-MAXAGREE method.

Hier-SUMSCORE Method: This method is an extension of the SUMSCORE method discussed in Section 5.2.3. For each datapoint, this method tries to maximize the sum of scores of selected labels, after penalizing for violation of class constraints. The scores from two views are combined through addition. There is a unique label assignment vector for each datapoint across two

views computed by solving the mixed integer program given Equation 6.2 (Figure 6.2 (a)).

Hier-PRODSCORE Method: This method is an extension of the PRODSCORE method. The mixed integer program for this method is very similar to that of the Hier-SUMSCORE method except that in the objective function, scores from two views are combined using product instead of addition of scores. There is a unique label assignment vector for each datapoint across two views.

Hier-MAXAGREE Method: This is an immediate extension of the MAXAGREE method described in Section 5.2.3, to incorporate class hierarchy. Different views can assign a datapoint to different sets of clusters. There is a penalty on cluster assignments being different across views, and any violation of the class constraints. Equation 6.3 (Figure 6.2 (b)) shows the mixed integer linear program to be solved for each datapoint. For each datapoint $s_i^{(1)}$, $s_i^{(2)}$ represent scores of x belonging to cluster i; and $y_i^{(1)}$, $y_i^{(2)}$ represent cluster assignment of x in view 1 and view 2 respectively. d_i is the penalty on $y_i^{(1)}$ and $y_i^{(2)}$ being different. $\zeta_{ij}^{(1)}$ and $\zeta_{ij}^{(2)}$ are the penalty terms for *Subset* constraints. Similarly, $\delta_{ij}^{(1)}$ and $\delta_{ij}^{(2)}$ are the penalty terms for *Mutex* constraints. α_1, α_2 and α_3 are constants that define relative importance of terms in the objective function.

Integer linear programming is used by several existing approaches to do constrained inference. [95] used mixed integer linear programming techniques to encode ontological constraints –they assign multiple plausible KB categories to 'emerging' entities, which are not yet represented in the KB, and consider mutual exclusion constraints as a post-processing step, so as to output a consistent set of category assignments. On the other hand, our method jointly models multi-view, subset and mutual exclusion constraints, within an iterative semi-supervised EM algorithm. Note that the hierarchical methods discussed here can work with any class ontology and are not limited to tree structured ontologies.

6.2.2 Experimental Results

Note that out of the eight multi-view datasets we experimented with in Section 5.3 only NELL_mv dataset is hierarchical. To test hierarchical multi-view methods, we performed experiments on the NELL_mv dataset with the training percentage varying from 5% to 30%. We generated 10 random train/test partitions for each training percentage value. Both flat and hierarchical methods got exactly the same train and test instances as input. Hierarchical methods had access to the entire hierarchy of labels and classified each datapoint w.r.t all classes in the hierarchy following the class constraints. Flat methods on the other hand learnt models only for the leaf classes of the hierarchy. We compared average F1 scores of all methods on the leaf classes. Further, we also compare these methods in terms of their average rank according to macro averaged F1 scores. According to this metric, the Hier-MAXAGREE method is the best followed by Hier-SUMSCORE, and Hier-PRODSCORE.

Table 6.2 shows the macro-averaged F1 results for all methods varying the training percentage. We can see that all the proposed hierarchical methods Hier-SUMSCORE, Hier-PRODSCORE and Hier-MAXAGREE improve over their flat counterparts in terms of macro-averaged F1 scores for all values of the training percentage. Columns marked as Δ shows the percentage relative

improvement of the hierarchical method over its flat counterpart. We can see that all methods benefit from hierarchy and the maximum percentage improvements are 18.3% for SUMSCORE, 6.7% for PRODSCORE and 20.1% for the MAXAGREE method. Overall, the Hier-MAXAGREE method performs best when the amount of training data is very small, followed by Hier-SUMSCORE and Hier-PRODSCORE methods. Further Hier-PRODSCORE works best for higher values of training percentages.

Train Percentage	View Agreement	Baselines		Flat vs. hierarchical multi-view methods								
		V12	CO-TRAIN	SUM-SCORE	Hier-SUMSCORE		PROD-SCORE	Hier-PRODSCORE		MAX-AGREE	Hier-MAXAGREE	
					F1	Δ		F1	Δ		F1	Δ
5	0.41	41.3	42.4	43.5	47.4	+9.0%	43.9	44.4	+1.1%	42.2	**47.5**	+12.6%
10	0.45	42.5	44.6	46.8	51.9	+11.0%	50.8	50.8	+0.1%	45.6	**52.0**	+14.0%
15	0.49	45.9	47.9	49.1	55.3	+12.8%	54.3	55.0	+1.1%	48.2	**55.4**	+15.1%
20	0.51	47.8	50.0	51.8	59.2	+14.2%	57.2	59.0	+3.1%	51.0	**59.2**	+16.0%
25	0.54	49.3	51.3	52.8	61.0	+15.5%	59.9	**62.6**	+4.4%	52.0	61.1	+17.4%
30	0.57	50.7	53.0	54.0	63.8	+18.2%	61.9	**66.0**	+6.6%	53.3	63.9	+20.0%
Avg. rank		7.8	4.7	6.0	**2.3**		4.2	**2.5**		7.2	**1.3**	

Table 6.2: Comparison of hierarchical vs. flat multi-view methods in terms of % Macro averaged F1 on the NELL_mv dataset. Column **Δ** lists the percentage relative improvement of the hierarchical methods over their flat counterparts. Last row of the table shows average rank of each method in terms of both macro averaged F1 and micro averaged F1 scores. Top 3 method ranks are bold-faced.

Figure 6.3 shows the correlation between view agreement and % relative improvement w.r.t. the V12 method. From the trend and correlation coefficient values we can say that improvements of PRODSCORE and MAXAGREE w.r.t. V12 are positively correlated with agreement between the views. Further, with lower agreement rate MAXAGREE performs best, whereas with high values of the agreement rate PRODSCORE performs best.

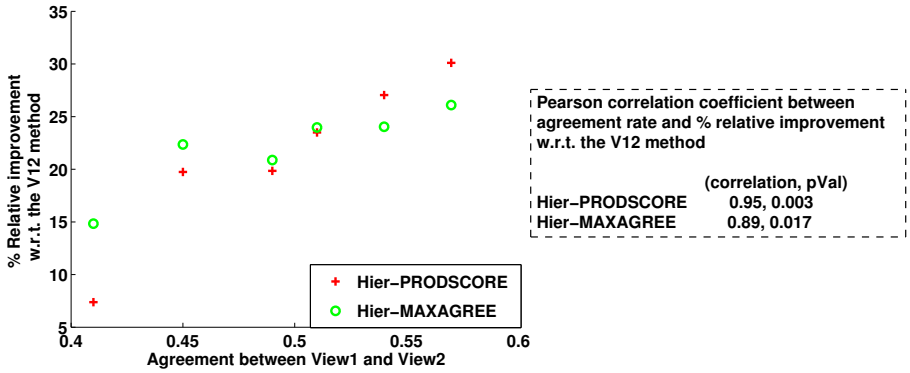


Figure 6.3: Percentage relative improvement of hierarchical multi-view methods for datasets with different view agreement rates.

Figure 6.4 compares the learning curves of hierarchical and flat versions of MAXAGREE and PRODSCORE methods vs. baselines. We can see all methods are improving with more training

data, however the learning curves of Hier-MAXAGREE and Hier- PRODSCORE are better than baselines V12 and COTRAIN, validating the effectiveness of proposed methods.

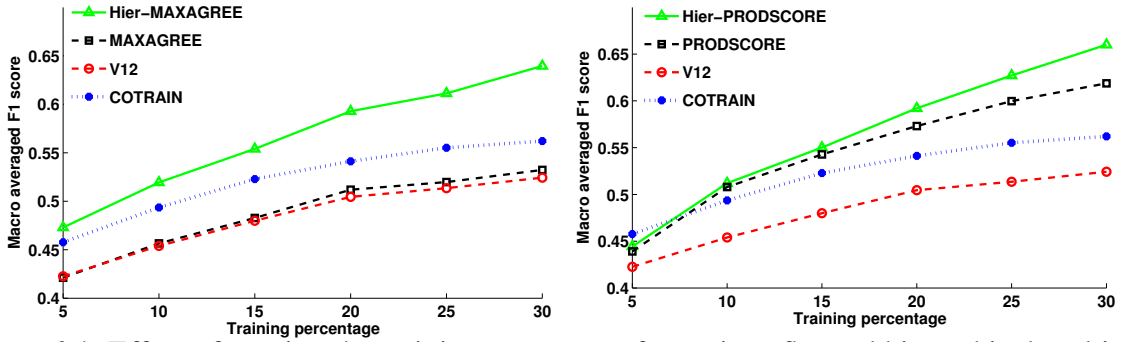


Figure 6.4: Effect of varying the training percentage for various flat and hierarchical multi-view methods on NELL_mv dataset.

Figure 6.5(a), (b) shows the scatter plot of hierarchical vs. flat methods (SUMSCORE, PRODSCORE and MAXAGREE) in terms of both macro and micro averaged F1 scores. Each method has six datapoints corresponding to the six different training percentages as shown in Table 6.2. We can see that hierarchical methods always outperform flat methods in terms of macro-averaged F1, but they might be worse in terms of micro averaged F1 scores. Figure 6.5(c) shows that the histogram of NELL_mv leaf class frequencies is skewed. It has been observed that skewed category distribution often leads to less reliable micro averaged performance [31] (since it is dominated by performance on the most popular classes). This can justify the surprising trend in Figure 6.5(a), (b) that for 6 out of 18 datapoints, the flat method outperforms the hierarchical method in terms of micro averaged F1 score. We found that all of these 6 datapoints come from PRODSCORE method. Hence Hier-MAXAGREE and Hier-SUMSCORE outperform their flat counterparts in terms of both macro and micro averaged F1 scores.

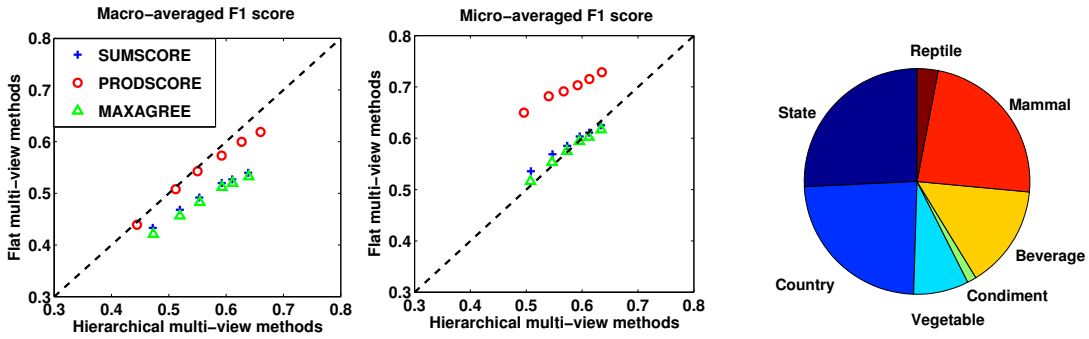


Figure 6.5: Results on NELL_mv dataset: (a) Scatter plot of hierarchical vs. corresponding flat multi-view methods in terms of macro-averaged F1 score, (b) Scatter plot in terms of micro-averaged F1 score, and (c) Class frequency histogram of leaf classes.

Finally we compare the flat and hierarchical multi-view methods in terms of average run-time. Figure 6.6 (a) shows the bar-chart of average run-times of methods. In our MATLAB implementation, the running time of proposed multi-view methods Hier-PRODSCORE, Hier-SUMSCORE and Hier-MAXAGREE are longer than traditional PRODSCORE, SUMSCORE

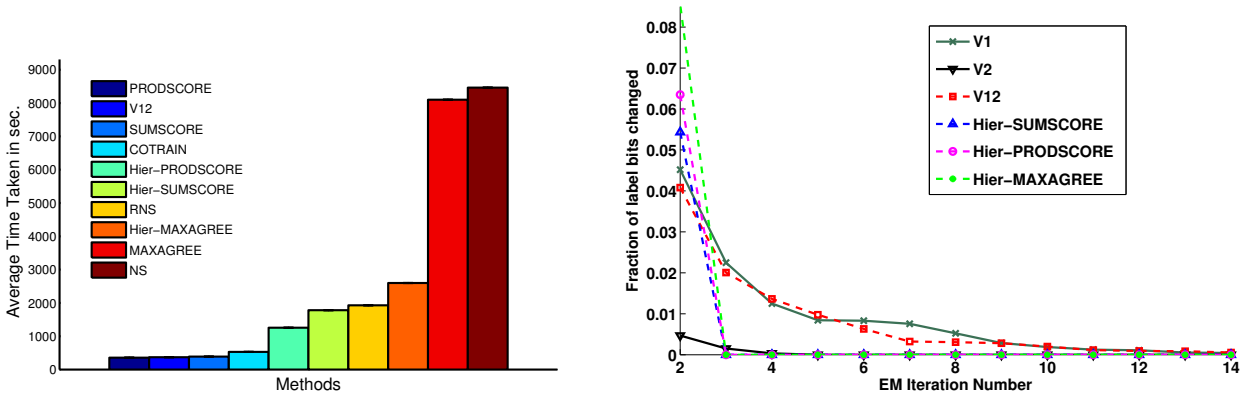


Figure 6.6: Results on NELL_mv dataset: (a) Average run-times of methods. (b) Convergence trends with 30% training data.

methods, but not unreasonably so. However, the flat multi-view method MAXAGREE is relatively more expensive, due to the combinatorial number of possible label assignments it needs to evaluate while solving mixed integer linear programs. We believe that adding the hierarchical constraints and ignoring unnecessary variables (an implementation trick [13]), reduces the number of possible candidate assignments to evaluate for Hier-MAXAGREE, making it more efficient than MAXAGREE. Further, Figure 6.6 (b) shows the convergence trends for some of the methods in terms of number of label bits flipped across EM iterations. The algorithms converge when no bit is flipped. We can see that hierarchical methods converge quickly compared to the V1 and V12 methods. Also note that V2 is the worse of two views, even though it converges quickly, its performance in terms of Macro averaged F1 score is lower.

Results in this section suggest that the superiority of hierarchical multi-view techniques based on our proposed optimization framework. It was also observed that with small amount of training data, Hier-MAXAGREE method gave state-of-the-art performance on NELL_mv dataset on the task of hierarchical multi-view learning in terms of both macro and micro averaged F1 scores.

6.3 Application: Automatic Gloss Finding for a KB using Ontological Constraints

In this section we will describe a real world application of the hierarchical semi-supervised learning algorithm (Algorithm 10) discussed earlier in this chapter.

While there has been much research on automatically constructing structured Knowledge Bases (KBs), most of it has focused on generating facts to populate the KB. However, a useful KB must go beyond facts. For example, glosses (short natural language definitions) have been found to be found to be very useful in other tasks such as Word Sense Disambiguation. However, the important problem of Automatic Gloss Finding, i.e., assigning glosses to entities in an initially

[13] If class A is disjoint with classes B and C, and C is subset of B, then $A \cap B = \phi$ and $C \subseteq B$ infer the remaining constraint $A \cap C = \phi$. We remove such redundant constraints while solving the optimization problem.

gloss-free KB is relatively unexplored – we address that gap in this section. In particular, we propose GLOFIN, a hierarchical semi-supervised learning system for this problem which makes effective use of limited amounts of supervision and available ontological constraints. To the best of our knowledge, GLOFIN is the first such system for this task. This work is published in the proceedings of WSDM 2015 [130].

Through extensive experiments on real-world datasets, we demonstrate GLOFIN's effectiveness. It is encouraging to see that GLOFIN outperforms other state-of-the-art SSL algorithms, especially when little labeled data is available. We also demonstrate GLOFIN's robustness to noise through experiments on two different KBs, a user contributed KB (Freebase) and an automatically constructed KB (NELL). To facilitate further research in this area, we have made datasets used in this chapter publicly available. Note that this is a larger scale task and we are looking at a single view dataset with a hierarchy of classes.

6.3.1 Motivation

Automatic construction of knowledge bases (KBs) has attracted much attention over the past years. Knowledge bases provide structured representation of entities and the relationships between them, which is key in semantic processing tasks such as concept tagging, disambiguation and normalization. While facts are obviously essential for a KB, a useful KB must contain more than facts. Many widely-used KBs, including Freebase and WordNet, also include *glosses*—i.e., natural language definitions of the entities in the KB. A large KB may contain glosses for each of the many entities it contains. In manually-constructed KBs like WordNet [90], glosses are typically provided as the KBs constructed by human experts; however, in many automatically generated KBs, like NELL [91] or YAGO [118] there are no glosses, or only a few. For instance, YAGO supports glosses, but only a small fraction of entities (68.9K out of 10M) have glosses assigned to them. Since manually adding definitions for a large number of entities is infeasible, glosses must be added automatically. In this section, we focus on the problem of augmenting an existing KB with gloss information. To the best of our knowledge, this is the first attempt of enriching an automatically-constructed KB, like NELL, with glosses.

To illustrate the problem, Figure 6.7 (top part) describes the structure of the NELL KB. As shown, the entities represented in NELL are organized in a hierarchy, in which nodes are linked over *is-a* (subset) relations. General semantic categories, like *fruit* and *company*, are represented by nodes that are linked to higher level categories in the hierarchy–*food* and *organization*, respectively. Concrete entity nodes, such as *Microsoft* and *Google*, are linked to their hypernym–*company*. In addition, entities are associated in NELL with their lexical aliases; e.g., 'Microsoft' and 'MS' are both aliases of *Microsoft*. Notably, lexical aliases are often ambiguous, e.g., the name 'Apple' refers to either *fruit:Apple* or *company:Apple*. While this representation of NELL KB provides valuable structured semantic knowledge, it does not contain glosses. Ideally, we would like to associate glosses to each entity node. For example, *Company:Apple* may be described as "Apple, formerly Apple computer Inc, is an American company headquartered in Cupertino...".

In addition to helping users understand the semantics (or intended semantics) of an entity, glosses are used in many technical tasks. In information retrieval, several recent approaches to query expansion make use of glosses [34, 129, 143] to improve the performance of ad-hoc infor-

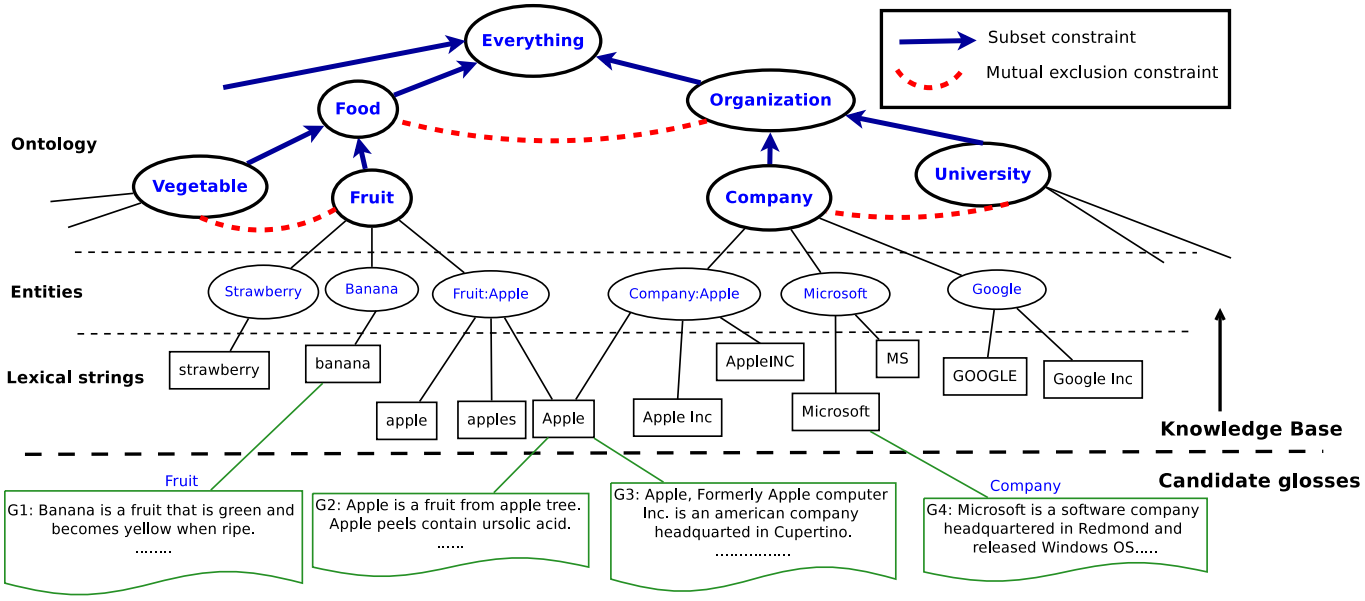


Figure 6.7: Graph construction for Entity Sense Disambiguation, using features like lexical matches, and is-a relationships (subset constraints) in the KB.

mation retrieval, based on a semantic, entity-rich representation of queries and documents. More generally, in the *entity linking* (EL) task [26, 65], named entity mentions in text are mapped onto canonicalized nodes in the KB, thus disambiguating the named entity mentions. Classical approaches to EL make heavy use of glosses: typically, an entity mention and surrounding context are treated as a query against a repository of glosses, with cosine similarity or word overlap [77] between the query and the node descriptions used for scoring, and the highest-scoring gloss being used to indicate the matching entity. For example, given a mention of 'Apple' in the context of 'Cupertino', a better match is to be expected against the gloss of *Company:Apple* (Fig. 6.7), compared with the alternative, *Fruit:Apple*.

One potential way of producing glosses might be to construct sentences out of the facts already stored in the KB: e.g., if the entity node *Michael Jordan* is linked to the category node *Professor* and related to the entity *UC/Berkeley* via the relationship *employedBy*, then it is possible to create a gloss that contains the sentence "Michael Jordan is a professor working at UC/Berkeley". However, many KB entities have few known relationships, so many such descriptions will be short and uninformative. Hence, rather than treating the task as a natural-language generation task, we consider an alternative method: we collect large numbers of definitional sentences, and attempt to match these existing sentences to KB entities. While the experiments described in this section focus on definitions collected from DBPedia, our approach is general, and could be applied as well to definitions harvested from any dictionary or glossary.

In short, the gloss finding task addressed in this section corresponds to matching potential glosses with the respective KB nodes. While some matches are obvious, many are ambiguous. While ambiguous matches are a problem in many other alignment tasks (e.g., the EL task described above), this task is unusual in that it is an *asymmetric* resource alignment, where a col-

lection of lexical glosses, which contain no structure information, is aligned against a structural KB, which contains no textual descriptions.

We define and address this task using semi-supervised learning (SSL) techniques. We rely on several simple intuitions to solve the above mentioned problem. First, it has already been observed [95, 118] that entity ambiguity is often resolved given its semantic category. Hence, rather than target a named entity linking problem, we solve an entity categorization problem. Next, we use the unambiguous glosses as labeled data for this entity categorization task. However, such labeled data is limited and noisy. We therefore propose **Gloss Finder (GLOFIN)** method for this task, which internally uses the semi-supervised Expectation Maximization (EM) method described in Algorithm 10.

6.3.2 Automatic Gloss Finding (GLOFIN)

In this section, we describe GLOFIN, our approach to the Gloss Finding problem. We first describe an entity-gloss candidate generation stage whose output is used by all the methods considered in the section.

Candidate Entity-Gloss Generation

In this step, for each gloss in the candidate pool of glosses given as input, we first identify the noun phrase that is being defined by the gloss. We refer to this noun phrase as *head-NP* in this section. For DBPedia abstracts this step is trivial in the sense that DBPedia dataset gives us the head-NP for each short abstract. However, we can easily apply existing syntactic taggers, e.g., the Stanford tagger [83], and detect the *head-NP* in the definitional sentence(s). After this, the head NP of each gloss is lexically matched against the entities in the KB. Even though simple string matchings are used for the experiments here, more sophisticated string matchers may also be used.

At the end of this stage, we end up with a set of candidate entity-gloss matchings. Please note that this relationship may be many-many and thereby ambiguous as one entity may be connected to multiple glosses, and a single gloss may be assigned multiple entities. Such ambiguity are resolved using the techniques described below.

Proposed Approach: Gloss Finder (GLOFIN)

Our Gloss Finder (GLOFIN) method uses the hierarchical semi-supervised Expectation Maximization (EM) algorithm (refer to Algorithm 10) to classify glosses into KB categories while satisfying ontological constraints. GLOFIN uses automatically acquired labeled data (unambiguous candidate glosses) and large amount of unlabeled data (rest of the candidate glosses) in an iterative EM framework to learn a model for this task. The hierarchical semi-supervised Expectation Maximization (EM) algorithm uses the MIP formulation (Equation 6.1) to get the best label vector for each datapoint.

Scaling GLOFIN

Note that, the MIP formulation presented in Equation 6.1 adds a constraint per subset and mutual exclusion constraints in the KB ontology. Further, in the E step of every iteration, we solve a MIP per gloss. These two result in longer runtimes for this method. We make our method scalable and efficient in following ways:

- We discard the redundant mutex constraints, the ones that can be inferred using remaining subset and mutex constraints.
- We reduce the number of classes considered per gloss, by keeping only top-Q classes relevant to the gloss, as detailed below.
- Since MIP for each gloss is independent, we parallelize the E step of every iteration, and consolidate results in the M-step.

Reducing the MIP size per gloss: We keep only a small number of categories in the optimization problem that is being solved for each gloss. We tried the following three ways of keeping only some of the candidate classes:

- Diameter of the class graph: The class graph is an undirected graph of ontology classes, where nodes are classes and edges represent either subset or mutual exclusion constraints. In this approximation, we rank all classes by the scores $P(C_j|X_i)$ for a datapoint X_i, and choose the top Q classes, where Q is the diameter of the category graph. Since the class assignments are hierarchical, we also include all ancestors of these top-Q classes in the optimization.
- Square-root of number of classes: Here, we select $Q = \sqrt{K}$, where K is the total number of classes in the ontology. Similar to diameter based method, we include all ancestors of these top-Q classes in the optimization.
- Thresholding: If the score $P(C_j|X_i)$ is greater than a predefined threshold then we consider the category. Note that the threshold is set for the entire dataset, so for each datapoint, it might result in a different number of categories and constraints.

6.3.3 Datasets and Experimental Methodology

In our experiments, we enrich two existing knowledge bases with available glosses, namely NELL and Freebase. NELL is a machine generated knowledge base that does not have existing glosses, whereas most Freebase entities have descriptions/glosses we can compare against. Our resource of glosses is DBPedia, a database of factual descriptions extracted from Wikipedia editions in 97 different languages. DBPedia contains a large set of Wikipedia titles and short abstracts.

A DBPedia short abstract of an entity is essentially the first paragraph (up to 500 characters) on the Wikipedia page of that entity. Some sample entity titles and their corresponding DBPedia abstracts are given in Table 6.3.

We now describe two experimental datasets that were labeled for evaluation purposes. We have made both these gloss finding datasets available[14] for the research community to help the

[14]The dataset can be downloaded from `http://rtw.ml.cmu.edu/wk/WebSets/glossFinding_wsdm_2015_online/index.html`

future research in this field.

Entity Title	Definition (DBPedia short abstract)
Platinum	Platinum is a chemical element with the chemical symbol Pt and an atomic number of 78. ...
Britwell	Britwell is a residential housing estate and civil parish in the north west of Slough Berkshire in the south of England. ...
Albi	Albi is a commune in southern France. It is the prefecture of the Tarn department. It is located on the River Tarn c 85 km ...
Nuraghe	The nuraghe is the main type of ancient megalithic edifice found in Sardinia developed during the Nuragic Age between 1900 ...
Cardassian	The Cardassians are an extraterrestrial species in the Star Trek science fiction franchise. First introduced in the 1991 Star Trek ...

Table 6.3: Sample candidate glosses.

Freebase dataset

Freebase is a tuple database used to structure general human knowledge. Currently Freebase consists of over 44M topics and 2613M facts about entities, their relationships and attributes. We use the Freebase entity snapshot provided by the ERD'14 challenge [26] as a gold standard mapping of Freebase to DBPedia abstracts. Thus we get a good quality training and test data for our semisupervised methods on the task of finding glosses for Freebase.

Table 6.4 includes detailed statistics of the Freebase dataset. Overall, Freebase entities in this dataset belong to 46 Freebase classes. Having considered their parent types, the dataset includes 66 classes in total. There are 5.5M Freebase entities in total that belong to these 66 classes. In order to obtain the underlying subset and mutual exclusion constraints, we built a co-occurrence matrix of entities belonging to each pair of classes. Based on this matrix, we elicited constraints of the two types: $|C_i \cap C_j| = 0$ implies that C_i is mutually exclusive with C_j; similarly, $|C_i \cap C_j| = |C_i|$ implies that C_i is subset of C_j. We discovered 46 subset constraints and 1,455 mutual exclusion class constraints from this data. Note that, Freebase is a sparse graph and hence the mutual exclusion constraints derived from this labeled dataset are approximate, and expected to be stricter than the ideal set of constraints.

NELL dataset

The NELL knowledge base is created by a machine learning system named Never Ending Language Learning [91]. NELL is composed of various information extraction components [24, 74, 137] that independently extract facts from text and semi-structured information on the Web. We used a version of the NELL KB that consists of 275 categories and 1.67M instances belonging to

Statistic	Dataset	
	Freebase	NELL
#Classes	66	275
#Subset class constraints	46	313
#Mutex class constraints	1455	18.1K
Diameter of class graph	4	10
$\sqrt{\#classes}$	9	17
#DBPedia abstracts	284.5K	246.7K
#Words	496.8K	472.4K
#(abstract, word) edges	5.7M	7.1M
#Unambiguous DBPedia abstracts	257.3K	195.8K
#Ambiguous DBPedia abstracts	32.8K	50.0K
#Ambiguous abstracts with ground-truth KB mappings	12.4K	383

Table 6.4: Statistics about the datasets.

these categories. Though the facts in NELL are extracted by a computer system, it takes as input a human created ontology containing categories, and subset and mutual exclusion constraints between them. So we don't have to infer class constraints as we did for the Freebase dataset.

We constructed a dataset using the NELL KB and candidate DBPedia abstracts. Statistics about the NELL dataset are included in Table 6.4. Unlike Freebase, NELL entities do not have glosses associated with them, so we do not have a ground truth gloss data for NELL. Further, since this KB is automatically generated by a computer system it contains noisy facts, hence the training data used by our methods is noisy.

Next we present results of two manual evaluation experiments for the NELL dataset. We do our first evaluation to understand the quality of automatically acquired seeds and predicted entity labels for ambiguous DBPedia abstracts. This will later help us to make claims about robustness of our proposed methods towards noisy training data.

Quality of automatically acquired seeds

Our methods consider unambiguous mappings of DBPedia abstracts onto NELL entities as labeled data. As such mappings are determined automatically for the NELL dataset, we performed manual evaluation in order to assess the quality of the unambiguous alignments, inspecting 100 randomly sampled unambiguous DBPedia-NELL matches. For each abstract in this sample, we evaluate whether the assigned class is *precise*, *correct*, and whether its higher level category is correct. We illustrate these measures using examples. If a DBPedia abstract about "Shively Field (public airport)" was mapped to the category "airport", then the mapping is considered to be precise, correct, and the higher level category is correct as well. Mapping between another DBPedia abstract about "Christopher McFarland (baseball player)" to the category "person-north-america", is judged as correct, but not precise, as there exists concrete "athlete" category in NELL. Finally, a mapping between "Jonathan Andersson (hockey player)" and "director", is incorrect and not precise. The higher level category in this case is correct however, since "director"

is a descendant of "person", which is correct.

Statistic	Value
#unambiguous abstracts evaluated	100
#abstracts s.t. assigned category was correct	81
#abstracts s.t. assigned category was most precise	72
#abstracts s.t. assigned higher level category was correct	94

Table 6.5: Evaluating quality of unambiguous mappings of DBPedia abstracts to NELL entities/categories.

Table 6.5 shows the results of this evaluation. We found that out of 100 unambiguous mappings between abstracts and NELL categories, 72% were precise and 81% were correct. The higher level category was correct in 94% of the considered examples. While these alignments are imperfect, we consider them to be of high quality. The experimental results described in the following section will show that our techniques make effective use of this automatically labeled data.

Manually creating gold-standard mappings from DBPedia abstracts to NELL entities

As we have already stated, NELL does not have glosses for its entities, and unlike Freebase there is no gold-standard mapping available from DBPedia abstracts to NELL entities. Hence we randomly sampled 383 DBPedia abstracts for which multiple candidate NELL entities and categories were found. For each abstract, we manually checked whether one of the candidate NELL entities and categories was a correct match. We also checked whether the most precise entity is present in the NELL KB. For some DBPedia abstracts, the most precise (entity, category) combination was not present in NELL, but the most precise category was present. The statistics of our evaluation are listed in Table 6.6.

Statistic	Value
#abstracts evaluated	383
%abstracts with at least 1 NELL entity, category match	79%
%abstracts with most precise entity candidate present in KB	68%
%abstracts with most precise category candidate present in KB	98%

Table 6.6: NELL Dataset: Manual evaluation of ambiguous glosses.

From Table 6.6 we can say that 79% of the DBPedia abstracts have at least one correct entity match in the candidate entity set. For example, a DBPedia abstract about "Michael Jordan" as Basketball player, can be matched to NELL entity "michael jordan:person"; however the most precise entity will be "michael jordan:athlete". We consider "michael jordan:person" as an accurate candidate entity, however "michael jordan:athlete' is the most precise entity, and "athlete" is the most precise category.

Methods

We experimented with the following eight methods for the gloss finding task. The first two methods are existing supervised and semisupervised methods.

- **SVM:** This is a traditional supervised linear SVM algorithm applied for the task of classifying head-NPs from candidate glosses into KB categories. We learnt a separate binary classifier for each KB category. For this purpose, we used the publicly available LibLinear package [43] with varying values of parameter "C" that controls penalty on slack variables. The higher the value of "C", the higher the penalty on slack variables, the harder the model will try to fit the training data. Similar method was used by Martinez et al. [84] which trains a decision list classifier on monosemous (unambiguous) words for the word sense disambiguation task. Instead we choose a more widely used supervised learning method SVM.
- **Label propagation:** In order to remove uncertainties from the candidate matchings generated in Section 6.3.2, we first use Modified Adsorption (MAD) [120], a representative graph-based semi-supervised learning (SSL) algorithm. This choice was prompted by the fact that such Label Propagation (LP) techniques have achieved considerable success in various weakly-supervised NLP tasks, and that they could potentially exploit the graph structure of the current problem.

  In order to apply MAD to this Gloss Finding problem, we first create an augmented version of the graph in Figure 6.7 by connecting each gloss to its content words. Please note that mutual exclusion constraints were not used in this augmented graph as MAD is not capable of handling such relations. We then inject each entity node with its own node-specific label. Such self-labeling approach was also used in [140], although for an entirely different problem. Starting with this augmented graph and self-injected seed labels, the MAD algorithm is used to classify the rest of the nodes in the graph. At the end of the algorithm, MAD assigns each candidate gloss a set of labels, where each label corresponds to an entity since the labels were entity-specific self-injections.

  This distribution of entities on a candidate gloss node is intersected with the candidate entities generated in Section 6.3.2, and the resulting entities are sorted based on the assignment score generated by MAD. The top entity in this ranked list is then chosen as the inferred entity match for the candidate gloss. As we shall see in Section 6.3.4, while this LP-based approach is effective in some settings, it is not always the case. To address this shortcoming, we present GLOFIN, our main proposed method is described below.
- **GLOFIN:** The next three methods are variants of our proposed GLOFIN algorithm (Section 6.3.2) using different document representations: GLOFIN-NB, GLOFIN-KM, and GLOFIN-VMF. These three document representations (Naive Bayes, K-Means, and von-Mises Fisher) are discussed in Section 4.1.3.

To evaluate the benefits of incorporating hierarchy, we also consider flat versions of semisupervised Naive Bayes, seeded spherical K-Means and seeded von-Mises Fisher. For each flat method, the E step (Algorithm 10: Line 8) computes $P(C_i|x_j)$ for all leaf classes C_i in the ontology and picks the one with maximum probability, skipping the mixed integer programming step (Algorithm 10: Line 9). The M step does not change. These methods are referred to as

GLOFIN-flat-NB, GLOFIN-flat-KM and GLOFIN-flat-VMF.

Features

Each gloss can be represented by the term frequency (TF) of words that occurred in it. We can also collect the inverse document frequency (IDF) of each word in the vocabulary from the entire gloss corpus. This gives us a choice of using either TF (mere term frequencies) or TFIDF (multiplication of TF and IDF values) feature representations. We use the TFIDF values to threshold the words used for each gloss. For all the experiments presented here, we keep only those words per gloss that cross TFIDF threshold[15] of 1E-3, thereby discarding stop-words and other unimportant words.

For each method, we performed experiments with both TF and TFIDF based feature representations (considering only those words that pass TFIDF threshold). In our experiments we found that SVM and Label-propagation methods give better performance for the TFIDF based feature representation, whereas all semisupervised EM methods (GLOFIN variants) work best with a TF based feature representation. Henceforth, we present all results using these choices of feature representations.

6.3.4 Experimental Results

Gloss Finding Results

In this section we present experimental results on the gloss finding task. For the first set of results we use all unambiguous glosses (glosses with only one candidate entity mapping to the KB) as training data. The next set of experiments make use of only 10% of unambiguous matches as training data. We compare all eight methods that we presented in Section 6.3.2: SVM, label propagation, the hierarchical and flat variants of GLOFIN.

For the SVM method, we tried different values of 'C' for the LibLinear package. We observed the change in performance varying the values of 'C' in the range [1E-24, 100] for both datasets. The change in performance for different values of C for both datasets, is plotted in Figure 6.8. We found that the best performance is achieved for 'C' in the range [1E-6, 0.01]. The results presented here are with C = 1E-6.

From the experiments presented here, we try to answer the following research questions:

- How does our proposed GLOFIN-NB method compare with SVM and label propagation on the task of assigning glosses for NELL and Freebase datasets?
- How do the Naive Bayes, K-Means and von Mises-Fisher variants of GLOFIN method compare?
- Does the use of ontological constraints always improve results for GLOFIN method?
- What is the effect of the different strategies of limiting the set of considered classes per gloss on scalability (and performance) of GLOFIN method?

[15] After this filtering step, each candidate gloss in the NELL dataset has on average 30 unique words.

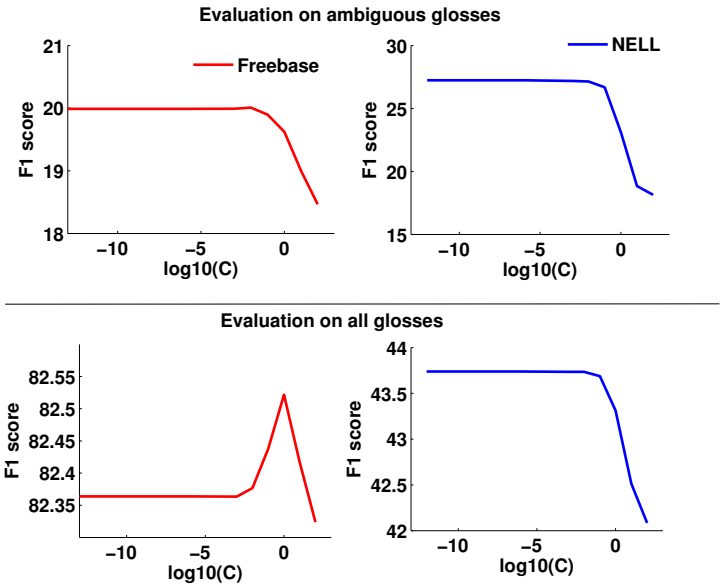


Figure 6.8: Evaluation of linear SVM with varying values of parameter 'C', for both NELL and Freebase datasets.

Here we compare our proposed GLOFIN-NB method against supervised SVM and semi-supervised label propagation methods. We compare them in two settings: first using all unambiguous glosses as training data and second, by simulating a harder problem i.e., using only 10% of the unambiguous glosses for training.

Comparing GLOFIN-NB against SVM and label propagation

Using all unambiguous glosses for training: Table 6.7 shows the summary of results while using all unambiguous glosses as training data and ambiguous glosses for evaluation.

We can see that the supervised SVM classification performs worst on both datasets. Label propagation gives the best F1 score for Freebase, however it performs poorly on the NELL dataset, whereas the GLOFIN-NB method works well on both NELL and Freebase datasets. On the Freebase dataset, GLOFIN-NB has higher precision, lower recall, and hence slightly lower F1 score than the label propagation method.

Both label propagation and GLOFIN-NB perform better on the Freebase dataset compared to the NELL dataset. This can be explained by the the difference in quality of the automatically acquired seeds for the two datasets. Note that the unambiguous glosses used for the Freebase dataset are from the ERD'14 gold standard dataset hence are very accurate, whereas for the NELL dataset our evaluation in Table 6.5 shows that only 81% of the seed gloss mappings were accurate. Hence the results in Table 6.7 indicate that our proposed GLOFIN-NB method is relatively robust to noise compared to the label propagation method.

We also investigated why the supervised SVM method performs poorly even though 80% of the total data is used as training. We found that there is a huge skew in the class frequencies. For the Freebase dataset with 45 leaf classes, the average number of training instances per class are

Method	Performance on "Ambiguous glosses"					
	NELL			Freebase		
	P	R	F1	P	R	F1
SVM	59.3	21.3	31.3	87.8	13.0	22.7
Label Propagation	42.8	54.0	47.8	89.8	**89.1**	**89.4**
GLOFIN-NB	**70.4**	**65.4**	**67.8**	**94.6**	74.2	83.2

Table 6.7: Comparison of gloss finding methods using all unambiguous glosses as training data and ambiguous glosses as test data. Best values in each column are bold-faced. GLOFIN-NB method is robust to noisy training data for the NELL dataset.

2.27K with a standard deviation of 10.8K; e.g., "/location/location" category has 72.6K training examples, whereas "/education/university" category has just 10 training examples. We observed that the number of seed examples are skewed across KB categories. Our method performs better than SVM due to two reasons, GLOFIN can deal with small amount of training data and we use unlabeled data in the process of EM to do better classification.

Note that in the experiments presented in Table 6.7, the number of unambiguous glosses used as training data covered a large part of the overall dataset (80% for NELL and 90% for Freebase). However, in real life scenarios, the amount of training data can be much smaller. The Freebase dataset is artificially created using ERD'14 data and a subset of Freebase KB categories. NELL populates its KB by bootstrapping against many different categories simultaneously, beginning with a small set of hand-chosen seeds. It may be the case that this learning process tends to favor entity names that can be confidently assigned to a single category (i.e., that are not highly ambiguous); to support this conjecture, we note that an alternative instance of NELL which used a larger set of automatically generated low-quality seeds required modifications to suppress the use of ambiguous entities in bootstrapping [94]. This suggests that other real-world KBs may have a lower proportion of unambiguous entities. Next, we conduct experiments which simulate this setting, by using only 10% of the available unambiguous matches as seeds.

Using 10% unambiguous glosses for training: Table 6.8 shows detailed results of GLOFIN-NB compared to SVM and label propagation methods when only 10% unambiguous glosses are used as training data. Since we are using only a fraction of unambiguous glosses as training data, the rest of them can be used as test data. Additionally, all gold-standard mappings of ambiguous glosses are always part of the test data. We generate 10 random train/test partitions and average results on these 10 runs. Performance on ambiguous glosses is particularly interesting hence listed separately. Note that "All glosses" include ambiguous as well as unambiguous glosses.

Method	NELL Dataset						Freebase Dataset					
	Ambiguous glosses			All glosses			Ambiguous glosses			All glosses		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
SVM	64.2	17.3	27.3	99.9	27.9	43.7	87.8	11.3	20.1	97.9	71.1	82.5
Label Propagation	55.0	12.3	20.1	99.7	5.2	9.8	84.6	27.5	41.5	99.7	88.1	**93.6**
GLOFIN-NB	71.7	62.0	**66.5**	99.9	62.0	**76.5**	95.1	72.0	**82.0**	97.6	79.6	87.6

Table 6.8: Comparison of gloss finding methods with 10% unambiguous abstracts used as training data. GLOFIN-NB always gives best or near-best F1 scores.

We can see that with less training data the performance of all methods degrades to varying degrees. Also, all methods give higher performance on "all glosses" compared to the harder task of matching ambiguous glosses. In terms of F1 scores on ambiguous glosses, SVM results in the worst performance and GLOFIN-NB method gives the best performance. In terms of F1 score on "all glosses", hier-NB performs better in all cases, except the Freebase dataset "all glosses" case where hier-NB has higher precision and lower recall, and hence lower F1 score compared to label propagation method.

To summarize, we showed that GLOFIN performs best on the harder task of matching ambiguous glosses when the amount of training data is small.

Discussion: Comparing the results across Tables 6.7 and 6.8, we can hypothesize that, with large fraction (90% for Freebase dataset) of glosses being unambiguous in our artificially created dataset, its very easy to get high values for F1 score using the label propagation method (Table 6.7), because rest of the 10% nodes might get high quality labels from its nearest neighbors. However when the amount of training data is smaller, generative models like hier-NB work better (Table 6.8) as they generalize better. Note that the performance of GLOFIN method on the NELL dataset saturates at the F1 score of 67.8% (from table 6.7), this might be due to the fact that the seed data is only 81% accurate.

One more advantage of GLOFIN lies in its generative nature. Label propagation is a transductive approach, whereas GLOFIN is a generative approach, i.e., EM models learnt by GLOFIN on a set of datapoints can be applied to an unseen datapoint having similar vocabulary. In other words, they can predict labels for an unseen datapoint. However, label propagation just predicts labels for the datapoints in the set being used while learning.

Comparing Variants of GLOFIN

Table 6.9 shows detailed results of our proposed methods (GLOFIN-NB, GLOFIN-KM and GLOFIN-VMF) w.r.t their flat counterparts (GLOFIN-flat-NB, GLOFIN-flat-KM and GLOFIN-flat-VMF). The GLOFIN-NB method works best on both NELL and Freebase datasets.

Method	Performance on "Ambiguous glosses"					
	NELL			Freebase		
	P	R	F1	P	R	F1
GLOFIN-flat-KM	74.8	36.5	49.1	97.6	61.0	75.1
GLOFIN-KM	65.2	49.8+	56.5+	96.1	71.8+	82.2+
GLOFIN-flat-VMF	73.6	44.5	55.5	96.3	56.0	70.8
GLOFIN-VMF	77.2	59.5+	67.2+	95.7	59.3+	73.2+
GLOFIN-flat-NB	70.3	59.1	64.3	95.9	71.1	81.7
GLOFIN-NB	70.4	65.4+	**67.8**+	94.6	74.2+	**83.2**+

Table 6.9: Comparison of GLOFIN variants using all unambiguous glosses as training data and ambiguous glosses as test data. Best values of F1 for each dataset is bold-faced. '+' in front of a hierarchical method score indicates that the score improved over its flat version.

GLOFIN-NB outperforms GLOFIN-KM in the experiments. Importantly, GLOFIN-NB models class priors. As mentioned before, the class frequencies in our datasets are very skewed;

hence modeling class priors is highly useful. In contrast, GLOFIN-KM merely computes cosine similarity of a datapoint with respect to a cluster centroid, hence its inferior performance. The GLOFIN-VMF method provides the second-best performance to GLOFIN-NB on the ambiguous gloss. Although GLOFIN-VMF is cluster-based, it also models class priors.

Effect of using ontological constraints

From Table 6.9, we can also see that all hierarchical semisupervised EM methods are better than their flat counterparts. Hence we conclude that using ontological constraints help improve gloss finding performance. Note that relative improvements of hierarchical methods are higher for the NELL dataset (upto 21% relative improvement in F1 scores). The reason traces back to evaluation of NELL seeds in Section 6.3.3, Table 6.5. We saw that 81% the leaf category seed labels were correct, whereas 94% of the higher level category labels were correct. Thus learning separate model parameters for higher level categories in the ontology and using ontological constraints to resolve ambiguity employed by hierarchical GLOFIN methods prove beneficial. Since Freebase seed labels are accurate, and the hierarchy contains just 2 levels, hierarchical models do not have as much added advantage over flat models as the NELL dataset, resulting in 9% relative improvement in F1 scores.

Comparing different ways of scaling GLOFIN

We tried various approximations to reduce the runtime of GLOFIN (Please refer to Section 6.3.2 for details). The summary is presented in Table 6.10. We found that setting Q, the number of classes considered in a MIP (Equation 6.1) as the diameter of the class graph, gives huge time savings and does not harm the performance in terms of F1 score on ambiguous entities. Hence we use this approximation for all runs of GLOFIN in this section. Other approximations like $Q = \sqrt{K}$ and the score threshold are also effective for maintaining goof F1 scores, however they were not as effective as their diameter method for lowering runtimes. Note that for the NELL dataset these approximations are crucial. Otherwise, run time on a PC machine (with 30GB memory) exceeds 64 hours. Due to large processing requirements, we do not have F1 score values for this case. In addition to this, we also use 12 parallel threads in the E step to compute label assignments of all datapoints in parallel.

Approximations	Performance on "Ambiguous glosses"			
	NELL		Freebase	
	F1	Time (in seconds)	F1	Time (in seconds)
keep all classes	-	>230.4K	80.4	15.2K
Q = diameter	81.7	5.2K	81.8	5.7K
Q = $\sqrt{K}$	83.4	15.9K	83.1	3.6K
score threshold= 1E-5	83.4	22.1K	71.4	9.3K

Table 6.10: Comparison of different approximations to scale our GLOFIN-NB method using 10% unambiguous glosses as training and ambiguous glosses as test data. Time measurements are in seconds.

Evaluating NELL to Freebase mappings via common glosses

One of the consequences of running GLOFIN on the NELL and Freebase datasets is that we get some mappings from NELL to Freebase entities based on common gloss matchings. Here we manually evaluate whether these mappings look meaningful. From the output of the GLOFIN-NB method, we randomly sampled 100 DBPedia abstracts that got assigned entities from both NELL and Freebase KB. Then we did a manual evaluation whether entities from NELL and Freebase correspond to each other, and whether the categories they belong to in respective KBs are semantically similar.

Eval Type	Statistic	Value
#Abstracts	Evaluated	100
	Assigned entities are correct, corresponding categories are semantically related	93
	Assigned NELL category is precise	92
	Assigned Freebase category is precise	38
#Category pairs	Found in 100 abstracts evaluated	39
	NELL category = Freebase category	6
	NELL category $\subset$ Freebase category	23
	Freebase category $\subset$ NELL category	1

Table 6.11: Evaluating quality of NELL to Freebase mappings via common DBPedia abstracts.

From Table 6.11, we can see that out of 100 abstracts that were evaluated, 93 of them had correct NELL and Freebase entities assigned to them, and their corresponding categories were semantically related. For 92 of those abstracts, the NELL category was precisely correct, while for 38 of them Freebase category was precise. This is due to the fact that the Freebase categories we use to build our dataset are more general categories like "/organization/organization", "/location/location", and more precise categories like "/location/city" and "/organization/biotechcompany" are missing. NELL has all these categories at higher granularity, hence it can classify DBPedia abstracts into more fine-grained categories. For instance, a DBPedia abstract about "Biotronik" is classified into the "biotechcompany" category from NELL, and the "/organization /organization" category from Freebase. We evaluate the entities from 2 KBs to be correct, and the corresponding categories as semantically related.

We also evaluated whether we can come up with a relationship between the categories corresponding to NELL and Freebase mappings. From 100 abstracts, we found 39 category pairs that got manually evaluated. We found six category pairs are equivalent. For Example, we marked the category "geopoliticallocation" from NELL to be equivalent to category "/location/location" from Freebase. In 23 category pairs, we found that a NELL category was strict subset of Freebase category, e.g., the "actor" category from NELL is strict subset of the "/person/person" category from Freebase. Only one category in Freebase "/broadcast/broadcast was found to be a strict subset of the NELL category "company". For nine category pairs we could not define an equality or subset relation between them. They include either the semantically unrelated class pairs like "visualartist" and "/location/location" corresponding to incorrect entity matches, and or semantically related categories like "drug" and "/business/brand". (There are many drugs like Donnatal, Excedrin that are classified as "drugs" in NELL and as "/business/brand" in Freebase. Though

these categories are related to each other we can not easily classify them into equivalence or subset relations.)

Example glosses matched by our method

Here we present some sample outputs of our methods. Table 6.12 lists some of the candidate glosses matched to NELL entities by our GLOFIN-NB method.

head-NP	Gloss	NELL entity selected by GLOFIN-NB	Candidate NELL entities
Kingston upon Hull	Kingston upon Hull frequently referred to as Hull is a city and unitary authority area in the ceremonial county of the East Riding of Yorkshire England It stands on the River Hull at its junction with ...	city:E	city:E, visualartist:E
Robert Southey	Robert Southey was an English poet of the Romantic school one of the so called Lake Poets and Poet Laureate for 30 years from 1813 to his death in 1843 Although his fame has been long eclipsed by that of his contemporaries and friends William Wordsworth ...	person-europe:E	personafrica:E, personeurope:E, politicianus:E
McGill University	McGill University is a research university located in Montreal Quebec Canada Founded in 1821 during the British colonial era the university bears the name of James McGill a prominent Montreal merchant ...	university:E	sportsteam:E, university:E
Andie MacDowell	Rosalie Anderson Andie MacDowell born April 21 1958 is an American model and actress She has received the Goldene Kamera ...	celebrity:E	celebrity:E, comedian:E, director:E
WLVW	WLVW is a radio station broadcasting a Christian music format Licensed to Salisbury Maryland USA the station is currently owned by Educational Media Foundation	radio-station:E	radiostation:E, televisionstation:E
Kevin Youkilis	Kevin Edmund Youkilis also known as Youk is an American professional baseball third baseman and first baseman for the New York Yankees of Major League Baseball A native of Cincinnati Ohio ...	athlete:E	athlete:E, person-canada:E
Caesionidae	The fusiliers are a family Caesionidae of fishes in the order Perciformes They are related to the snappers but adapted for feeding on plankton rather than on larger prey They are found at reefs ...	fish:E	fish:E, mollusk:E
Chloroper-lidae	Chloroperlidae is a family of stoneflies commonly known as green stoneflies There are more than 180 species in the family They appear in colors of green and yellow	insect:E	insect:E, mollusk:E
Evolutionism	Evolutionism refers to the biological concept of evolution specifically to a widely held 19th century belief that organisms are intrinsically bound to increase in complexity The belief was extended to ...	visualart-movement:E	religion:E, visualart-movement:E
Royal Bank of Canada	The Royal Bank of Canada RBC French Banque Royale du Canada RBC Royal Bank or RBC Financial Group is the largest financial institution in Canada as measured by deposits revenues and market capitalization ...	bank:E	bank:E, credit-union:E
Neurology	Neurology from Greek neuron nerve the suffix logia study of is a medical specialty dealing with disorders of the nervous system To be specific neurology deals with the diagnosis and treatment of ...	academic-field:E	academicfield:E, medicalprocedure:E

Table 6.12: Example outputs produced by GLOFIN-NB on the NELL dataset. (Due to space restrictions, repeatitions of Head-NP in each table row are replaced by 'E'.)

6.4 Related Work

6.4.1 Hierarchical Semi-supervised Learning

There has been some work on modifying EM to incorporate side information such as multi-view constraints or domain knowledge based constraints on outputs. The posterior regularization framework [49] modifies the EM algorithm to encode domain constraints on the expected output, which has also been extended to incorporate soft agreement in a multi-view setting [50]. Our technique Hier-MAXAGREE is different in that it maximizes agreement between the labels produced in each view while satisfying ontological class constraints if available, whereas the method proposed by Ganchev et al. [50] minimizes the difference between actual scores produced by the views.

Chang et al. [29] proposed an extension of EM algorithm for encoding the task specific domain knowledge base constraints on the output. Our hierarchical techniques are similar to this in the sense that we also modify the EM algorithm to do constrained learning. However our techniques are different from [29] in the sense we are using constraints specific to a dataset: e.g., for the NELL_mv dataset, the ontological constraints we used are the same as the ones which were used while building the NELL knowledge base. We did not need any hand curated constraints for the specific task that we worked on. Further we take care of the agreement across views and ontological constraints in a single framework using multiple soft constraints.

Graph based multi-view semisupervised learning techniques have also been proposed in past few years. Wang et al. [135] proposed a multi-graph based semisupervised learning technique that can incorporate multiple modalities, and multiple distance functions in the task of video annotation. Lee et al. [75] proposed a new graph-based multi-label propagation technique and applied it to large datasets by utilizing a map-reduce framework. Though these methods can handle multi-view data, they fail to address the scenarios where classes/labels are arranged in a hierarchy and inference needs to be done following certain constraints between these classes.

Here we propose a mixed integer programming based optimization method that can deal with any class hierarchy defined by sets of subset and mutual exclusion constraints. Interestingly, a recent work has used mixed integer LP techniques to encode ontological constraints [95]–they assign multiple plausible KB categories to 'emerging' entities, which are not yet represented in the KB, and consider mutual exclusion constraints as a post-processing step, so as to output a consistent set of category assignments. In contrast, we apply subset as well as mutual exclusion constraints, within an iterative EM algorithm, using semi-supervised learning Finally, while the application of earlier variants of the EM-based framework focused on the named entity categorization task, we use the proposed methods for addressing the novel task of generating glosses for an existing KB.

6.4.2 Gloss Finding

Glosses are an important resource of word meanings, which has been used by word sense disambiguation (WSD) [96] algorithms for decades. For example, the popular Lesk method [77] and its variants [8, 71] infer a word's sense by measuring the overlap between available context words and the glosses of candidate senses. Traditionally, WordNet [46] has been used as the

main resource of word senses and respective glosses. Several recent works have investigated the problem of automatic gloss extraction, so as to improve coverage in specific knowledge domains. Duan and Yates [39] constructed sense descriptions for selected keywords given unlabeled documents in the Biomedical domain, and used these descriptions to perform WSD. Others [36, 44] constructed domain-specific glossaries using definitional sentences extracted from the Web, and successfully used these glossaries for domain WSD. We share their motivation for obtaining glosses, however rather than generate these glosses, our goal is to associate available glosses with an existing KB.

We match given glosses against the respective entity nodes in the KB. This task is closely related to Entity Linking (EL), an entity-focused variant of the WSD problem [93], where named entity mentions in a given text are linked with the respective nodes in a KB. Advanced EL methods consider multiple entity mentions in a document collectively in order to enrich the contextual information that is modeled per entity [58, 134]. Such methods are ineffective for short text fragments, like search queries, or glosses. In fact, we face a 'chicken and egg' problem, as most EL methods assume that KB entities have glosses, whereas we do not [60, 80]. Here, we are interested in enriching a KB with glosses, so as to support entity linking and other downstream applications, such as ad-hoc information retrieval [34, 143].

Various works have addressed the problem of resource alignment, but considered different settings. The Yago ontology [118], for example, unifies Wikipedia and WordNet using heuristics that consider the structure of both resources. Ponzetto and Navigli [105] mapped Wikipedia articles onto WordNet synsets, using the textual descriptions of entity pairs. We rather consider an asymmetric setting, where the KB includes a semantic structure but lacks textual descriptions, and the aligned glosses are not assumed to be organized in any fashion. Similarly, a recent work [104] addressed the alignment of an arbitrary pair of lexical resources, including machine-readable dictionaries with no structure. They proposed to induce a network structure for dictionary resources, however textual similarity remains an important component in their approach. Importantly, having enriched a given KB with glosses using the proposed framework, this will facilitate its alignment and integration with other KBs [32].

We addressed the gloss finding problem using a semi-supervised framework. In particular, we consider label propagation (LP), where gloss nodes are being associated with candidate entity nodes based on their relatedness in a joint graph. Talukdar [119] discuss the use of weakly supervised label propagation methods in information extraction and integration. Previous work applied LP to perform WSD [146]. In that case, the graph modeled words in parallel documents in two languages, where word ambiguity in one language was resolved based on graph relatedness to their candidate translation in the other language, and vice versa. The use of graph-based methods is highly prevalent in WSD literature. In particular, random walks over a semantic graph, like WordNet, have shown to yield state-of-the-art results [4, 105]. Notably, linking nodes in the WordNet graph based on lexical gloss overlaps has been shown to contribute significantly to the performance of graph-based WSD [4].

6.5 Conclusion

In this chapter, we showed that our proposed linear programming based formulation can be easily extended to multi-label classification and can incorporate hierarchical class constraints. For the NELL dataset with hierarchy of classes, our extended optimization methods produced better results when compared to flat multi-view clustering baselines in terms of macro-averaged F1 score. An interesting direction for future research is to extend the techniques to the problem of multi-view Exploratory Learning[128] which does multi-view semi-supervised learning in the presence of unanticipated classes. Such techniques can further be used for populating knowledge bases along with discovering new classes from multi-view unlabeled data.

Further we applied this technique for the important but relatively unexplored problem of Automatic Gloss Identification, i.e., automatically generating gloss (short definitional sentences) for an initially gloss-free knowledge base (KB) by matching candidate glosses to entities in the KB. To the best of our knowledge, this is the first such system for this task. GLOFIN employs a hierarchical clustering algorithm that internally uses unambiguous DBPedia abstracts as seeds and the KB ontology as constraints to match an ambiguous candidate gloss to the right entity from KB. GLOFIN use mixed integer programming formulation to assign the most likely set of labels for each gloss, while following the class constraints posed by the KB ontology.

We present experiments using NELL and Freebase as KBs and DBPedia abstracts as candidate glosses. Our quantitative and qualitative evaluations show that GLOFIN is effective and that it outperforms other state-of-the-art algorithms like label propagation and Support Vector Machines (SVM). We also demonstrate GLOFIN's robustness to noise through experiments on a wide variety of KBs, ranging from user contributed (e.g., Freebase) to automatically constructed (e.g., NELL). To facilitate further research in this area, we have already made datasets used in this section publicly available, and we plan to release the code into the public domain. In future, we would like to extend GLOFIN to discover new entities and new categories to further enrich the knowledge bases.

6.6 Future Directions

The LP based formulation discussed here works in the semi-supervised setting. A natural direction of future work is to make this technique exploratory, i.e., to dynamically add new clusters of datapoints that do not belong to one of the known classes, while taking into consideration the ontological constraints. In the next chapter we describe our proposed approach for this task, named "Hierarchical Exploratory Learning".

Chapter 7

Exploratory Learning with An Incomplete Class Ontology

In an open-domain entity classification task, topic or concept hierarchies are often incomplete. In Chapter 4 we showed that in a non-hierarchical semi-supervised classification task, the presence of such unanticipated classes can cause semantic drift for seeded classes. Exploratory learning [128] method was proposed to solve this problem by exploring a different number of classes while learning classifiers for each of them. However it is limited to the flat classification task. This chapter extends the Exploratory learning method to hierarchical classification tasks. This work is partially published in the proceedings of AKBC 2013 [127] and the remaining work is under preparation for submission [124].

We experimented with subsets of the NELL [25] ontology and text, and HTML table datasets derived from the ClueWeb09 corpus. Our method OptDAC-ExploreEM improves seed class F1 on average by 13% when compared to its semi-supervised counterpart (OptDAC). It also outperforms both previously proposed exploratory learning approaches FLAT-ExploreEM and DAC-ExploreEM [127] in terms of seed class F1 on average by 10% and 7% respectively.

7.1 Motivation

A common way to organize information is by classification into a concept or topic hierarchy. The Open Directory Project and the Yahoo! Directory are examples of topic hierarchies developed to organize pages on the Web. Wordnet, NELL and Freebase are examples of large knowledge bases that organize entities into concept hierarchies. However, in an open-domain task, hierarchies are often incomplete, in the sense that there is meaningful structure in the data not captured by the existing hierarchy. For example, consider a hierarchical entity classification task, with class hierarchy 'onto-1' shown in Figure 7.1 (left). There are 2 types of locations defined in it: 'State' and 'Country'. However, the unlabeled data about entities on the Web can include location entities of type 'City', 'Museum' etc. , that are absent in this ontology, hence called 'unanticipated classes'.

In Chapter 4 we showed that in a non-hierarchical semi-supervised classification task, the presence of such unanticipated classes hidden in the unlabeled data can cause semantic drift for

Figure 7.1: Subsets of the NELL [25] ontology used in our hierarchical classification experiments. Some nodes in onto-1 are bold-faced, they are used as seeded classes in the experiments described in Section 7.5.

seeded classes. They also proposed an approach to solve this semantic drift problem by employing exploratory learning. This algorithm is an extension of the semi-supervised EM algorithm that can create new classes for datapoints that do not belong to the classes known upfront. It explores different numbers of classes while learning. In this chapter we extend the exploratory learning algorithm for hierarchical classification tasks.

Our proposed OptDAC-ExploreEM method traverses the class hierarchy in top-down fashion to detect whether and where to add a new class for the given datapoint. It also uses a systematic optimization strategy to find the best set of labels for a datapoint given ontological constraints in the form of subset and mutual exclusion constraints.

We demonstrate OptDAC-ExploreEM's effectiveness through extensive experiments on datasets constructed with subsets of the NELL ontology (shown in Figure 7.1) and text and semi-structured HTML table datasets derived from the ClueWeb09 corpus. In particular, OptDAC-ExploreEM improves seed class F1 on average by 13% when compared to its semi-supervised counterpart. It also outperforms flat exploratory learning approach FLAT-ExploreEM in terms of seed class F1 on average by 10%.

7.2 Our Approach

Our method is derived from the Exploratory EM algorithm described in Chapter 4 (Algorithm 3). Exploratory EM is limited to flat class hierarchies. Here we propose the Hierarchical Exploratory EM algorithm which can work with incomplete class hierarchies and small amount of seed labels.

7.2.1 Problem Definition

Given a set of class constraints Z_k, a small set of labeled data points X^l, their labels Y^l, and a large set of unlabeled data points X^u; the task is to label data points from X^u, adding m new classes if needed and extending the class constraints to Z_{k+m}. Here, each point from X^l can have multiple labels at different levels of the hierarchy satisfying constraints Z_k, and Z_{k+m} defines the class constraints on k seed and m newly added classes, and the labels Y^u of X^u satisfy Z_{k+m}. Also note that when we add m new classes to the ontology, along with some new constraints, the old constraints should still get satisfied, i.e., $Z_k \subseteq Z_{k+m}$. Next let us see different methods proposed to solve this problem.

Algorithm 11 Generic Hierarchical Exploratory EM

function Hierarchical-ExploreEM (X^l, Y^l, X^u, Z_k): θ_{k+m}, Z_{k+m}, Y^u
Input: X^l labeled data points; Y^l labels of X^l;
X^u unlabeled data points;
Z_k manually input constraints on k seed classes $\Longleftarrow$
Output: $\{\theta_1, \ldots, \theta_{k+m}\}$ parameters for k seed and m newly added classes; Y^u labels for X^u;
Z_{k+m} Set of class constraints between $k+m$ classes; $\Longleftarrow$
{Initialize classifiers θ_j for class C_j using seeds provided for C_j}
$\theta_1^0 \ldots \theta_k^0 = argmax_\theta L(X^l, Y^l)$
while class assignments AND #classes not converged **do**
  k_{old} is #classes before E step. Log-likelihood *BaseLL* $= logP(X|\theta^{(t)}_{k_{old}}, Z^{(t)}_{k_{old}})$
  {E step: (Iteration t) Classify each datapoint at each level}
  for i=1 to $|X|$ **do**
    Find $P(C_j|X_i)$ for all classes C_j
    {Assign a bit vector of labels for each unlabeled datapoint. A new class gets created for a datapoint that does not fit into existing classes.}
    $Y_i^{(t)}$ = **ConsistentAssignment**($P(C_j|X_i)$, h, $Z^{(t)}$) $\Longleftarrow$
    {If a new class is created, then class constraints are updated accordingly.}
    $Z^{(t)}$ = **UpdateConstraints**($X^l, Y^l, X^u, Y^u, Z^{(t)}$) $\Longleftarrow$
  end for
  k_{new} is #classes after E step. Log-likelihood *ExploreLL* $= logP(X|\theta^{(t)}_{k_{new}}, Z^{(t)}_{k_{new}})$
  {M step: Recompute model parameters based on $Y^{(t)}$ }
  if Model selection criterion(k_{old}, *BaseLL*, k_{new}, *ExploreLL*) selects exploratory model **then**
    {Adopt the new model with k_{new} classes}
    $\theta^{(t+1)}_{k_{new}} = argmax_\theta L(X^l, Y^l, X^u, Y^{u(t)}|\theta^{(t)}_{k_{new}}, Z^{(t)}_{k_{new}})$
    $Z^{(t+1)} = Z^{(t)}_{k_{new}}$
  else
    {Keep the old model with k_{old} classes}
    $\theta^{(t+1)}_{k_{old}} = argmax_\theta L(X^l, Y^l, X^u, Y^{u(t)}|\theta^{(t)}_{k_{old}}, Z^{(t)}_{k_{old}})$
    $Z^{(t+1)} = Z^{(t)}_{k_{old}}$
  end if
end while
end function

7.2.2 Flat Exploratory EM

One simple way to use Exploratory EM algorithm [128] for our purpose will be to run it as it is at each level of hierarchy. Henceforth, we refer to this approach as FLAT-ExploreEM and consider it as a baseline for our proposed Hierarchical Exploratory EM approach. At each level, it selects one of the existing classes or creates a new class in case the datapoint doesn't clearly belong to one of the known classes. This algorithm does not make explicit use of class constraints while making class assignments at each step. Hence the assignments done by this algorithm might not be consistent, since assignments at level 3 are not influenced by assignment at level 2.

7.2.3 Hierarchical Exploratory EM

We propose a generic Hierarchical exploratory learning algorithm that can take a set of class constraints in terms of subclass or mutual exclusion constraints. This algorithm, in each iteration, assigns a bit vector to each data point with one bit per class. Class constraints decide whether a bit vector is consistent or not, e.g., if class constraints include "Car is of type Machine", then for consistency, when the bit for "Car" is set, the bit for "Machine" should also be set. Further new classes can be added during each iteration, hence the length of these bit vectors changes dynamically and the algorithm maintains class constraints containing old as well as newly added classes. The generic algorithm is described in Algorithm 11. There can be multiple ways to implement functions "ConsistentAssignment" and "UpdateConstraints".

Lines 13-19 of Algorithm 11 do model selection. Similar to Exploratory EM [128], at the end of every E step, we evaluate two models, one with and one without adding extra classes. These two models are scored using a model selection criterion like AIC, BIC or AICc, and the model with best penalized data likelihood score is selected in each iteration. The extended Akaike information criterion (AICc) suited best for our experiments since our datasets have large number of features and small number of data points. The class constraints are modeled in the same way as described in Section 6.1.1.

7.3 Variants of Hierarchical Exploratory EM

Hierarchical Exploratory EM (described in Algorithm 11) is a generic algorithm that can be instantiated in different ways, by changing the functions "ConsistentAssignment" and "UpdateConstraints". Next we describe two such variants namely, DAC-ExploreEM and OptDAC-ExploreEM.

7.3.1 Divide and Conquer (DAC-ExploreEM)

One simple instantiation of Algorithm 11 can be done by using Divide-and-Conquer (DAC) strategy introduced in [127]. Here we assume that classes are arranged in a tree hierarchy, and classes at any one level are mutually exclusive. To do class assignments for any unlabeled datapoint, we traverse the class ontology from root to leaf level. Every data point belongs to the root

node. Then at each level we chose the best label at that level and consider only its children as candidates at the next level.

Further we check whether the probability distribution among the candidate classes at each level is nearly uniform (using heuristic described in Algorithm 5) to decide whether to create a new class at that level. We do not describe the "ConsistentAssignment" and "UpdateConstraints" functions formally, however they can be easily derived by setting parameter $q = 1$ in the OptDAC-ExploreEM algorithm that we present next.

DAC-ExploreEMcan do semi-supervised learning if the presence of unanticipated classes. However, we will see in the experimental evaluation (refer to Section 7.5.2) that DAC-ExploreEM could provide marginal improvements over the baseline (Exploratory EM). During the error analysis we found that the classification at higher levels of hierarchy is not perfect, and once we make a decision at level i of the ontology, there is no way to change the decision once we move on to level $i+1$. Next we try a softer version of this method, that keeps track of q nodes at each level instead of keeping only the best label.

Also, note that the example ontologies in Figure 7.1 have tree structure. However, in practice, class constraints can be more complicated (e.g., overlapping classes can exist). The DAC-ExploreEM algorithm is limited to a tree structured ontology and assumes mutual exclusion of classes at any level of hierarchy. Next we present the OptDAC-ExploreEM algorithm that can work with more complicated class constraints.

7.3.2 Optimized Divide and Conquer (OptDAC-ExploreEM)

Algorithm 12 describes how the "ConsistentAssignment" and "UpdateConstraints" functions are implemented for this approach. It is similar to DAC-ExploreEM method in the sense that we traverse the classes in top down fashion, and check whether new class needs to be created at each level of the hierarchy. However, at each level l of the class hierarchy, a mixed-integer program is run to get optimal class assignments for the active classes from levels 1 to l. Further instead of considering only the best class, top-q classes from each level are selected to be added into the set of active classes which are used in turn to select the candidate classes at the next level of hierarchy.

This method combines the Divide-and-Conquer strategy to detect/place new classes and the mixed integer programming based optimization strategy to assign an optimal set of labels to a datapoint given the ontological constraints. The optimal label assignment method is generic enough to support any set of subset and mutual exclusion class constraints. The new class detection based on Divide and Conquer can be extended for non-tree hierarchies using a breadth first search strategy that can be applied to any graph. Hence the OptDAC-ExploreEM method can be extended easily for non-tree structured ontologies.

Note that similar to DAC-ExploreEM, it makes greedy decisions about new cluster creation. However, it performs overall optimization of label assignments while satisfying the ontological constraints. This lets us correct any sub-optimal decisions made by the greedy heuristic at higher levels of the class hierarchy. Further, as we increase the value of q, we might get some improvement in accuracy at the cost of increased runtime. Since the value of q directly decides the size of active nodes set used while taking the decision at each level of the hierarchy, there is a trade-off between time complexity and solution optimality. For all the experiments in this paper, we added

top two classes per level to the set of selected classes (i.e. $q = 2$) in Line 21 of Algorithm 12. This approach is referred to as OptDAC-ExploreEM below.

For each datapoint, the optimal label assignment given class constraints (Algorithm 12: Line 24) if computed using Equation 6.1. To solve these mixed integer linear programs we used the MOSEK solver [5]. We have used such optimization techniques for the task of semi-supervised learning in the presence of multiple data views and hierarchical class constraints [123, 130]. Here we use this formulation for the task of hierarchical exploratory learning.

Algorithm 12 OptDAC-ExploreEM

function ConsistentAssignment-OptDAC ($P(C_j|x)$, $Z_k\}$): Y_x, Z_{k+m}
Input: $P(C_j|x)$ probability distribution of classes given a datapoint x; Z_k class constraints on k seed classes.
Output: $label(x)$ assignment of x to classes satisfying Z_k;
Z_{k+m} extended set of class constraints on $k + m$ classes.
h is the height of the class ontology.
for l = 1 to h **do**
  if x has seed label at level l **then**
    label(x, $level_l$) = seed label;
  else
    $candidateC$ = children of the active classes
    if $candidateC$ is not empty **then**
      Let P_{cand} = probability distribution over $candidateC$
      if Is Nearly Uniform (P_{cand}) (using Algorithm **??**) **then**
        Create a new class C_{new} at level l
        Initialize parameters for class C_{new} using x
        Set parent(C_{new}) = class choice at level $l - 1$
        Add C_{new} to active classes
      end if
      P_{active} = probability distribution over active classes
      Z_{active} = class constraints between active classes
      Choose $label(x, level_l)$ by computing optimal label assignment considering (P_{active}, Z_{active}) (using Eq. 6.1)
      Add top-q classes to the set of active classes using P_{active} as scores
    end if
  end if
end for
end function
function UpdateConstraints-OptDAC (X, Y, Z^{old}): Z^{new}
Input: X: Dataset; Y: Class assignments for each point in X;
Z^{old}: Old constraints on the existing set of classes.
Output: Z^{new}: Updated set of class constraints,
Each newly created class is assigned a single parent at the time of creation
Add each parent, child relationship as a constraint in $Z_{k'}$
end function

7.4 Datasets and Experimental Methodology

In this section we present the experimental results of our Hierarchical Exploratory EM approach. Figure 7.1 shows the two ontologies that we used, each being a subset of NELL's ontology at different point in NELL's development.

Dataset	Corpus	Ontology	#Classes	#Levels	#Classes per level	#Entities	#Contexts	#(Entity, context) pairs	#(Entity, label) pairs
Text-Small	Text	onto-1	11	3	1, 3, 7	2.5K	3.4M	8.8M	7.2K
Text-Medium	Text	onto-2	39	4	1, 4, 24, 10	12.9K	6.7M	25.8M	42.2K
Table-Small	Tables	onto-1	11	3	1, 3, 7	4.3K	0.96M	6.3M	12.2K
Table-Medium	Tables	onto-2	39	4	1, 4, 24, 10	33.4K	2.2M	21.4M	126.1K

Table 7.1: Statistics of the hierarchical entity classification datasets used in this chapter.

7.4.1 Datasets

Our task includes discovering new classes that are not present in the input class ontology. To make the evaluation easier, we created datasets that have ground truth labels for all entities in them, i.e., the entire class hierarchy is known. However, only part of that hierarchy and corresponding training data is given as input to the algorithm. The rest of the classes and corresponding labels are unknown to the algorithm, and used only during evaluation. Thus we are simulating the scenarios where some classes are known while others are unanticipated. To achieve this, we derived four datasets using the two ontologies (shown in Figure 7.1) and two feature sets extracted from the ClueWeb09 corpus. The first ontology, named onto-1 in Figure 7.1 (left), has 3 levels and 11 classes. The second ontology, named onto-2 in Figure 7.1 (right), has 4 levels and 39 classes.

We created our datasets using the two corpora: *Text-Patterns* and *HTML-Tables*, both derived from ClueWeb09 data [23]. The *text-patterns* corpus contains frequency counts of text context patterns that occurred with each entity w.r.t.t̃ext sentences that appeared in ClueWeb09 dataset. The *HTML-tables* corpus contains frequency counts of table columns in which entities occurred, derived from HTML tables that appeared in ClueWeb09 dataset. For Example, an entity "Pittsburgh" having a *Text-Patterns* feature, value being ("lives in _arg1", 1000) means that the entity "Pittsburgh" appeared in arg1 position of the context "lives in _arg1" for 1000 times in the sentences from ClueWeb09 dataset. Similarly, an entity "Pittsburgh" having a *HTML-Tables* feature, value being ("clueweb09-en0011-94-04::2:1", 1) means that the entity "Pittsburgh" appeared once in HTML table 2, column 1 from ClueWeb09 document id "clueweb09-en0011-94-04".

To create a dataset from an ontology, we took all entities that are labeled with at least one of the classes under consideration, and retrieved their feature representation in terms of occurrences with text patterns or HTML table columns. Thus we created four datasets Text-Small to Table-Medium, using combinations of two ontologies and two corpora. Table 7.1 describes the statistics about these four datasets. We plan to make our hierarchical entity classification datasets publicly available.

7.4.2 Methods

We experimented with three different methods for the entity clustering task: FLAT, DAC and OptDAC. Each of them have EM and Exploratory EM variants. The EM variant performs semisupervised learning with the seeded classes, whereas the Exploratory EM variant can add extra classes discovered from unlabeled data.

- FLAT: This method performs flat entity classification at each level of the class hierarchy. Decisions made at each level are independent.
- DAC: This method performs hierarchical classification of entities, by making class assignment decision in top-down fashion. At each level maximum probable class is selected and candidates at next level are children of the class selected at previous level.
- OptDAC: This is another hierarchical classification method. It also makes class assignment decisions in top-down fashion. The creation of extra classes and placing them in hierarchy is similar to DAC method. However, class assignments at each level l are determined by solving a linear optimization problem considering all nodes under consideration ($ActiveC$ in Algorithm 12) spanning levels 1 to l. Hence the greedy decisions about new cluster creations are combined with overall optimization of label assignments while following ontological constraints.

Further we used seeded K-Means algorithm for clustering (as described in Section 7.2.3) and the MinMax criterion [128] for new class creation[16].

7.4.3 Methodology

Remember that for the experiments in this chapter, we feed a part of the ontology (seed classes) to the algorithm and rest of the part (unanticipated classes) is hidden from the algorithm. EM variant of each method learns classifiers only for seed classes. Along with this, Exploratory EM variant of each method can add new classes. To make the semi-supervised and exploratory variants of each method comparable, we use "macro-averaged seed class F1" as the evaluation metric. It is computed by macro averaging F1 values of seed classes only. Further, if the Exploratory EM variant improves the seed class F1 over the EM variant, it indicates that adding extra classes helped towards keeping the seed classes pure (i.e., reducing semantic drift).

This "macro-averaged seed class F1" metric is further averaged for 10 runs of each algorithm. Each run's input consists of different seed ontologies and randomly sampled 10% of relevant datapoints as seed examples. The same set of inputs is given to all algorithms being compared. Note that, for a given dataset with a choice of of seed classes and training percentage, there are many ways to generate a train-test partition. We report results using 10 random train-test partitions of each dataset. The same partitions are used to run all the algorithms being compared and to compute the statistical significance of the results.

For Text-Small and Table-Small, we generated 10 sets of seed examples, for the same seed ontology. The chosen seed ontology is bold-faced in Figure 7.1 (left). Here seed ontology always contains the same 7 out of 11 classes. For Text-Medium and Table-Medium, seed ontology

[16]MinMax criterion introduces a new class for a data point and posterior distribution of classes given the data point, if the maximum to minimum probability ratio is less than 2.

also varies across runs. In each run, we randomly chose 10 leaf nodes according to their class frequency (i.e. popular class is more likely to be chosen in each run). After sampling 10 leaf nodes (sampling without replacement), we included all their ancestors to create the seed ontology for that run. The average ontology size generated using this method was 16.7. Table 7.3 column 2 shows avg. number of seed classes chosen at each level of the hierarchy. 10% of the datapoints from the leaf classes are then randomly sampled, and hierarchical labels of these datapoints are used as training data for the run.

7.5 Experimental Results

In this section we will compare semi-supervised and exploratory variants of the FLAT, DAC and OptDAC methods, in order to answer certain research questions.

7.5.1 Do ontological constraints help?

Table 7.2 shows the comparison between semi-supervised versions of both methods. The best values in each row (per dataset per level in the hierarchy) are bold-faced. We can see that for every row, the best performance was given by a method that uses ontological constraints while clustering. Hence we can conclude that using ontological constraints do help.

Dataset	Level	Macro-avg. Seed Class F1		
		W/o constraints	W constraints	
		FLAT	**DAC**	**OptDAC**
		EM	EM	EM
Text-Small	2	46.6	47.1	**52.0**
	3	23.5	**26.1**	25.8 △
Text-Medium	2	53.2	**53.7**	53.3
	3	27.9	33.4 ▲	**33.9** ▲
	4	17.4	24.5	**26.8** △
Table-Small	2	69.5	74.6 ▲	**74.8** ▲
	3	36.8	38.8 △	**38.9** ▲
Table-Medium	2	62.7	**64.8**	62.2
	3	43.7	46.4 ▲	**48.0** ▲
	4	47.3	**57.7** ▲	57.1 ▲

Table 7.2: Comparison of FLAT, DAC, and OptDAC methods in the semi-supervised setting. ▲ (and △) indicates that improvements of the DAC and OptDAC methods are statistically significant w.r.t the FLAT method with 0.05 (and 0.1) significance level.

We also did statistical significance tests with 0.05 (and 0.1) significance level denoted by ▲ (and △) in Table 7.2. Results show that in 5 out of 10 cases the gains of DAC over FLAT are statistically significant. Whereas the OptDAC method proved to be better by producing statistically significant gains over FLAT in 7 out of 10 cases.

7.5.2 Is Exploratory learning better than semi-supervised learning for seed classes?

We present the comparison of Semi-supervised vs. Exploratory versions of all three methods in Table 7.3. The best performance in each row is bold-faced. From Table 7.3 we can see that, the Exploratory EM version of each algorithm improves seed class F1 when compared to EM for all three methods in 24/30 cases. Our proposed approach OptDAC-ExploreEM improves seed class F1 on average by 13% when compared to its semi-supervised counterpart. While comparing among the Exploratory EM approaches, the OptDAC method independently beats previously proposed FLAT and DAC methods in 8/10 cases each. Empirically, OptDAC-ExploreEM outperforms FLAT-ExploreEM and DAC-ExploreEM in terms of seed class F1 on average by 10% and 7% respectively.

Dataset	#Seed /#Ideal classes	Level	Macro-avg. Seed Class F1					
			FLAT		**DAC**		**OptDAC**	
			EM	Exploratory EM	EM	Exploratory EM	EM	Exploratory EM
Text-Small	2/3	2	46.6	**64.4**	47.1	61.8	52.0	62.6
	4/7	3	23.5	32.7	26.1	36.3	25.8	**42.3** ▲
Text-Medium	3.9/4	2	53.2	50.2	**53.7**	47.2	53.3	52.5
	9.4/24	3	27.9	27.0	33.4	26.8	33.9	**34.9** ▲
	2.4/10	4	17.4	25.8	24.5	29.4	26.8	**31.6** ▲
Table-Small	2/3	2	69.5	75.8	74.6	76.2	74.8	**80.0** ▲
	4/7	3	36.8	**43.9**	38.8	40.9 △	38.9	41.5
Table-Medium	3.9/4	2	62.7	61.3	64.8	**65.0** ▲	62.2	**65.0** ▲
	9.4/24	3	43.7	49.1	46.4	48.6	48.0	**50.1**
	2.4/10	4	47.3	52.2	57.7	**59.9** ▲	57.1	58.4 ▲

Table 7.3: Comparison of FLAT, DAC, MIP, and OptDAC methods using KM representation on Text-Small to Table-Medium. ▲ (and △) indicates that improvements of the DAC-ExploreEM and OptDAC-ExploreEM methods are statistically significant w.r.t FLAT-ExploreEM method with 0.05 (and 0.1) significance level.

Further we did statistical significance tests for performance improvements of DAC-ExploreEM and OptDAC-ExploreEM over FLAT-ExploreEM, with 0.05 (and 0.1) significance level denoted by ▲ (and △) in Table 7.3. It shows that improvements of DAC-ExploreEM are statistically significant in 3/10 cases, whereas improvements of OptDAC-ExploreEM are significant in 6/10 cases. Thus OptDAC-ExploreEM turns out to be the best method being compared.

7.5.3 What is the effect of varying training percentage?

We ran the OptDAC-ExploreEM method on datasets Text-Small and Table-Small with different values of training percentage averaged over 10 random train/test partitions of our data. In Figure 7.2, we can see that as training percentage increases the performance of OptDAC-ExploreEM method improves, especially as we go down the hierarchy (relative improvements are more at Level 3 compared to Level 2).

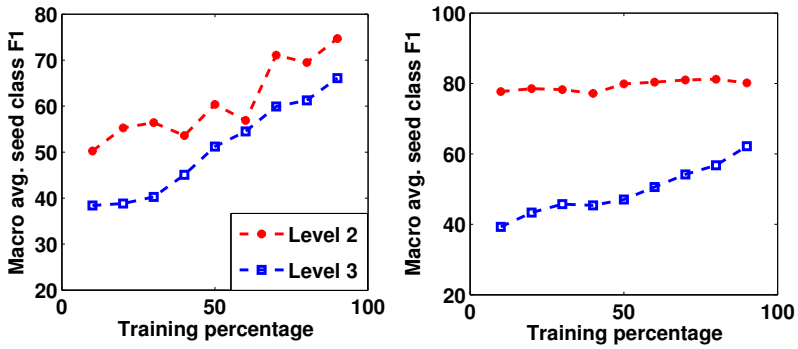


Figure 7.2: Comparison of OptDAC-ExploreEM method with different training percentage on datasets Text-Small (left) and Table-Small (right).

7.5.4 How do the methods compare in terms of runtime?

Here we compare runtimes of all methods averaged over all the runs with different seed ontologies and seed training data. In our MATLAB implementation, the running time of Exploratory EM is much longer. Table **??** shows that the increase in runtime of Exploratory EM variants w.r.t. their EM counterparts is by the factor 3.2 on average across all methods and datasets. Further the exploratory methods take on average 10.3 times when compared to EM variant of the FLAT method.

In particular, OptDAC-ExploreEM method is on average twice as expensive as DAC-ExploreEM. This is due to the fact that DAC-ExploreEM takes greedy decisions at each level of the hierarchy, whereas OptDAC-ExploreEM keeps track of the top-q active nodes. We set $q = 2$ in these experiments. Thus the value of q results in a trade-off between improvement in seed class F1 and increased runtime of the algorithm.

7.5.5 Evaluation of extended cluster hierarchies

In this section, we present the evaluation of extra clusters added by our Hierarchical Exploratory EM algorithm to the incomplete class ontology given as input. If it started with k classes, and produced $k + m$ classes as output, we first need to label these m extra classes. Since our datasets are completely labeled, each new class can be assigned a majority label based on entities assigned to that class and level of the class in the hierarchy. For Example, if the class is at level 2 in the hierarchy then we choose the best label at level 2 of the class hierarchy.

Figure 7.3 shows an example of an extended class hierarchy generated by OptDAC-ExploreEM algorithm on Table-Small starting with 7 seed classes from onto-1. The bold-faced nodes are from the seed ontology, and nodes in blue (non bold-faced) are the ones added by our proposed exploratory learning methods. It is labeled using the above described majority label approach. (Note that there are multiple clusters with same label, to differentiate them we have labeled them as "Food_1", "Food_2" etc.)

Once this labeling is done, we can measure the precision of labels across (parent, child) links in the cluster hierarchy. For Example, in Figure 7.3, parent-child link between (Food_2,

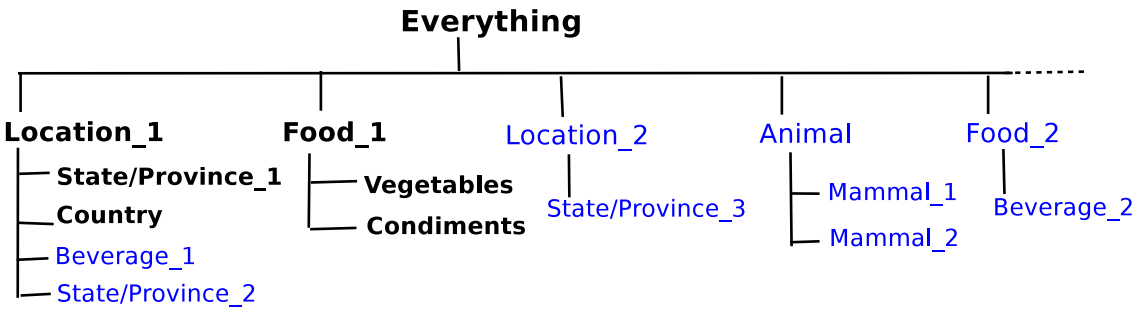


Figure 7.3: An example extended ontology applying our OptDAC-ExploreEM method on the Table-Small dataset. The seeded classes are bold-faced, whereas the newly added classes are in Blue and are not bold-faced.

Beverage_2) is correct, however the link between (Location, Beverage) is incorrect. These links can be classified into seed or extra links based on whether the child cluster was one of the seed clusters or introduced by the Hierarchical Exploratory EM algorithm.

Dataset	%Precision of (parent, child) edges for clusters		
	Seed	Extra	All
Text-Small	88.3	84.5	85.5
Text-Medium	80.7	22.6	46.8
Table-Small	100.0	90.4	92.4
Table-Medium	95.7	36.1	63.9

Table 7.4: Precision of child, parent edges created by OptDAC-ExploreEM.

Table 7.4 shows the link precision values for OptDAC-ExploreEM algorithm when run on all four datasets. We can see that for seed clusters, accuracy numbers are high (81 - 100%) for all four datasets. In terms of extra clusters, for the datasets Text-Small and Table-Small with simpler ontology "onto-1", and unanticipated class fraction being low, the precision of edges for extra clusters is very high in around 85%. Whereas for Text-Medium and Table-Medium, with more complicated ontology "onto-2", and with higher fraction of unanticipated classes, the precision for extra clusters is quite low around 30%. This indicates that the task of detecting new classes and adding them at right positions in the ontology is a challenging task, and it gets even more challenging with the complexity and degree of incompleteness of the input ontology.

7.6 Related Work

There has also been some work in unsupervised hierarchical clustering [11, 82, 141] that can discover cluster/topic hierarchies given a large unlabeled dataset, however they do not make use of any supervision that might be available. Exploratory learning differs in that we learn the number of clusters as well as centroids for those clusters jointly in a single run of the EM

algorithm, while using the available supervision for seed clusters. Apart from standard K-Means, our EM framework can also be used with other clustering/classification algorithms like Naive Bayes and von Mises-Fisher, and we specifically evaluate the performance difference on the seeded classes.

There has been a lot of research done in the area of supervised hierarchical classification [21, 55, 142]. These methods assume that the class hierarchy is complete and there is enough training data to learn classifiers for each node in the class hierarchy. On the other hand we considered the situation where only part of the ontology is known upfront with very few seed examples for each of the seed class. Further our method can be easily extended to cases where class constraints are more complicated than the examples considered in this chapter, e.g., overlapping classes and mutual exclusion constraints. Another related research area is constructing web-scale knowledge bases [2, 25] by doing information extraction from various data-sources. NELL internally uses coupled semi-supervised learning [25] that takes into account subclass and mutual exclusion constraints among classes to filter extraction patterns and instances at the end of each bootstrapping iteration. The ontology (class hierarchy) is not explicitly used in the prediction process. I.e. it does flat classification with class constraints applied as post-processing in between two iterations of bootstrapping. Our approach on the other hand does collective hierarchical classification.

There has also been some work to extend existing ontologies. Mohamed et al. [92] propose a co-clustering based two step approach to discover new relations between two already existing classes in the knowledge base. These new relations are named using centroid features of the intermediate clusters. This method is focused on relation discovery between known classes. Snow et al. [116] discover new WordNet synsets by using evidence from multiple sources, however their approach is focused on discovering new isA relations, and not meant for building hierarchical classifiers for the learnt hierarchy. Pal et al. [99] proposed a technique based on Indian Buffet Process that could learn with existing feature hierarchies as well as extend them based on structure discovered from unlabeled data. Their method relies only on the containment relationships and the hierarchies they experimented with are domain specific e.g., restaurants domain.

Reisinger and Paşca [108] addressed the same problem as ours, working with the Wordnet hierarchy. Their fixed-structure and sense selective approaches use the Wordnet hierarchy directly and annotate existing concepts with generic property fields (attributes). On the other hand, the Nested Chinese Restaurant Process (nCRP) approach is hierarchical extension of LDA to infer arbitrary fixed-depth tree structures from data. nCRP generates its own annotated hierarchy whose concept nodes do not necessarily correspond to Wordnet concepts. Our method is in the middle of these two approaches, as it uses the existing class hierarchy with small amount of training data and extends it dynamically as new clusters of datapoints are discovered.

Another set of techniques focus on completely unsupervised information extraction and ontology discovery [12, 53, 108, 126]. Though very effective, these approaches are not making use of the valuable information hidden in the existing knowledge bases. Our approach is relatively novel in the sense that it is in between semi-supervised and unsupervised learning, where some part of ontology is known, and this knowledge is used to discover the missing parts of the ontology along with populating it with new data instances.

To define the similarity among two entities we use bag of word features about co-occurrences of text patterns with the noun-phrases. There can be more effective approaches of document

representation like the lower dimensional continuous Skip-gram features, proposed by Mikolov et al. [89]. Their technique learns low dimensional vectors that potentially embed semantics of noun-phrases.

7.7 Conclusions

We propose the Hierarchical Exploratory EM approach that can take an incomplete seed ontology as input, along with a few examples of each seed class, to populate new instances of seed classes and extend the ontology with newly discovered classes. Experiments with subsets of the NELL ontology and and text, semi-structured HTML table datasets derived from the ClueWeb09 corpus show encouraging results in terms of seed class F1 scores.

Our proposed hierarchical exploratory EM method, named OptDAC-ExploreEM performs better than flat classification and hierarchical semi-supervised EM methods at all levels of hierarchy, especially as we go further down the hierarchy. Experiments show that OptDAC-ExploreEM outperforms its semi-supervised variant on average by 13% in terms of seed class F1 scores. It also outperforms both previously proposed exploratory learning approaches FLAT-ExploreEM and DAC-ExploreEM in terms of seed class F1 on average by 10% and 7% respectively.

7.8 Future Research Directions

In the future, we would like to apply our method on datasets with larger and non-tree structured class hierarchies. We briefly discussed how our proposed OptDAC-ExploreEM method can be used for this task. Further, our experiments focused on an information extraction task of classifying entities into a knowledge base class hierarchy. However, our techniques can also be used for other tasks like document classification into topic hierarchies on datasets like Reuters, and classifying images into a class hierarchy for datasets like ImageNet.

Chapter 8

Concluding Remarks

Here, we first summarize the overall contributions, then we list the observations from experiments presented in different chapters of this thesis. Finally, we also present the potential future directions for this research.

8.1 Thesis Summary

We proposed several extensions to the traditional semi-supervised learning algorithm using Expectation Maximization. Combining all these extensions leads to a generic semi-supervised learning framework that can incorporate 1) the presence of unanticipated classes, 2) multiple data views, and 3) ontological class constraints. This framework helps reduce semantic drift on known seeded classes as well as provides a single method that solves a continuum of tasks from supervised learning to clustering. In both flat and hierarchical classification/clustering settings, we observed that running an algorithm in an exploratory setting gave comparable or better seed class F1 score when compared to the traditional semi-supervised counterpart [124, 127, 128].

Further, exploratory learning is more powerful in conjunction with ontological class constraints which can be imposed as soft constraints on the label vectors assigned to a datapoint. We also found that considering ontological constraints while inferring labels improves overall performance compared to flat classification for the the entity classification and gloss finding tasks [124, 127, 130]. Through experiments in this thesis, we showed that a single semi-supervised learning framework could handle NLP tasks like entity/document classification, entity/word sense disambiguation, gloss finding and ontology extension. Code for the flat exploratory K-Means, GLOFIN algorithm and most of the datasets we created during this research can be downloaded from `http://rtw.ml.cmu.edu/wk/WebSets/exploratory_learning/`.

Contributions

In this thesis we first proposed exploratory learning techniques (Chapter 4) that improve the robustness of semi-supervised learning methods to the presence of unanticipated classes in the unlabeled data, thereby reducing semantic drift of seeded classes. Exploratory EM introduces new classes on-the-fly during learning based on the intuition that hard-to-classify examples—specifically, examples with a nearly-uniform posterior class distribution—should be assigned to

new classes. The exploratory versions of these SSL methods often obtained dramatically better performance—e.g., on the Delicious_Sports dataset up to 90% improvements in F1, on the 20-Newsgroups dataset up to 27% improvements in F1, and on the Reuters dataset up to 200% improvements in F1.

In Chapter 5, we investigated the problem of semi-supervised learning in the presence of multiple data views. We formulated the problem as an optimization problem, and solved it using the standard EM framework. We then focused on the sub-problem of assigning labels to each datapoint (part of the E step), and studied various methods for such prediction. Our proposed method solves a mixed integer linear program to find consistent class assignments given the scores in each data view. Because our multi-view techniques are broadly similar to co-training based algorithms, we also compared them to a seeded version of the multi-view spherical K-Means algorithm that is proposed by Bickel and Scheffer [10]. Our methods produced better performance in terms of macro-averaged F1 score compared to this co-training baseline.

Next, we worked on incorporating ontological class constraints (Chapter 6) such as subset and mutual exclusion. For a dataset with hierarchy of classes, our extended optimization methods produced better results when compared to flat semi-supervised learning baselines in terms of macro-averaged F1 score. We further applied this method for the important but relatively unexplored problem of Automatic Gloss Identification, i.e., automatically generating gloss (short definitional sentences) for an initially gloss-free knowledge base (KB) by matching candidate glosses to entities in the KB. To the best of our knowledge, this is the first such system for this task. Our method, GLOFIN, employs hierarchical clustering algorithm that internally uses unambiguous DBPedia abstracts as seeds and the KB ontology as constraints to match an ambiguous candidate gloss to the right entity from KB. Our quantitative and qualitative evaluations using NELL and Freebase KBs show that GLOFIN is effective and that it outperforms other state-of-the-art algorithms like label propagation and Support Vector Machines (SVM). We also demonstrate GLOFIN's robustness to noise in the seed data.

Finally in Chapter 7, we combined the exploratory learning and constrained learning techniques into the Hierarchical Exploratory EM approach that can take do exploratory learning in the presence of an incomplete class ontology, i.e., it can populate more instances of known classes, and extend the ontology with newly discovered classes in a single learning framework. Our proposed method, named OptDAC-ExploreEM, performs better than flat classification and hierarchical semi-supervised EM methods at all levels of the class hierarchy, especially as we go further down the hierarchy. Experiments show that OptDAC-ExploreEM outperforms its semi-supervised variant on average by 13% in terms of seed class F1 scores. It also outperforms both previously proposed exploratory learning approaches FLAT-ExploreEM and DAC-ExploreEM in terms of seed class F1 on average by 10% and 7% respectively.

Exploratory Knowledge Acquisition with Human in the Loop

The exploratory learning techniques proposed in Chapter 4 and 7, proposes a way to do exploratory knowledge acquisition. The overall idea is represented in Figure 8.1. Traditional knowledge acquisition can add new facts for categories present in the ontology. In addition to this, the exploratory learning techniques can suggest new categories that can be added to the ontology. A human can look at the new category suggestions, along with instances of these new

categories, to pick the ones that look meaningful.

This kind of learning mechanism is in between open domain knowledge extraction and closed domain (ontology driven) knowledge extraction, i.e., it can make use of the available training data and at the same time discover new categories from unlabeled data.

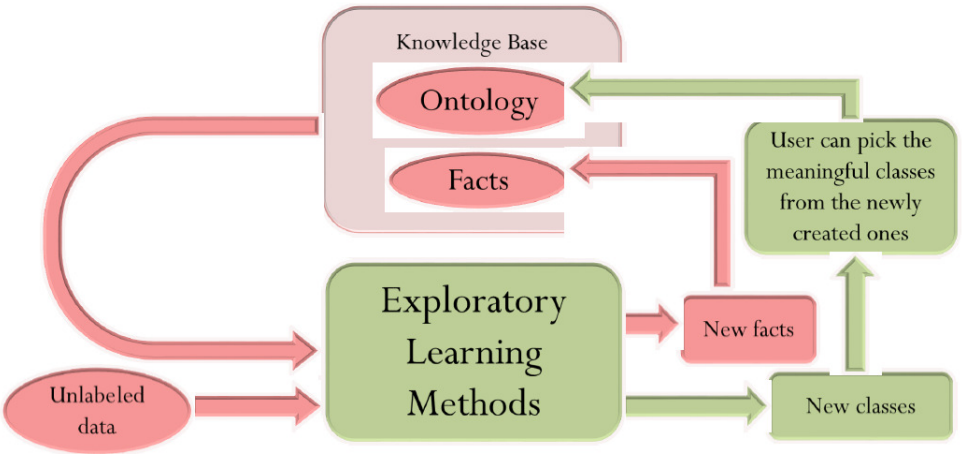


Figure 8.1: Exploratory Knowledge Acquisition with Human in the Loop.

8.2 Future Research Directions

This thesis opens up many interesting directions for future research. A most straightforward future work can be to understand how well our exploratory learning techniques work for datasets where the classes are not clearly separable (overlapping topics). It would also be interesting to analyze what fraction of classes need to be seeded to be able to discover the rest of them from unlabeled data.

The intuitions behind our techniques can be used to extend few other techniques to make them more effective. Further, the techniques proposed in this thesis are not limited to the tasks presented in this thesis. Next, we explain some of these future directions in terms of advancements in terms of the techniques and applications.

New Techniques:

Naming the Newly Discovered Concepts

Apart from reducing the semantic drift of seeded classes, another important advantage of exploratory learning methods is to discover multiple clusters of datapoints that did not fit any of the classes given to the algorithm. These are potentially new concepts that can be added to an existing knowledge base. For example, in our experiments with the WebSets table corpus and NELL KB, we found that with some manual inspection, our system could add good quality concepts like “music scales”, “dental procedures” etc., that were missing in the NELL KB.

Automatically naming the newly discovered clusters of entities is another interesting direction of future research. In Chapter 3, we described one such technique, i.e. those hypernyms that occurred with the candidate entities in the context of Hearst patterns were used as candidate labels for newly discovered clusters. Another way to achieve this could be to describe a cluster by the centroid for the cluster. For Example, for a newly discovered cluster of news stories, with a bag of words feature representation and TFIDF weights, a centroid feature vector will give us the top distinguishing words that occurred most frequently in the cluster.

Exploratory Extensions of Mixed-Membership Methods

Our proposed Exploratory Learning technique [128] shows that the near uniformity heuristic if applied to Chinese Restaurant Process, gives performance gains in terms of seed class F1 and running times (Chapter 4). Exploratory EM in its current form is a hard-EM algorithm, i.e., in the E step each datapoint gets assigned to only one of the clusters/classes. It would be an interesting direction of future research to build mixed-membership exploratory learning techniques, i.e., a soft-EM version of our algorithm, and use modified CRP process, to induce new clusters.

New Applications:

Word Sense Disambiguation using WordNet Hierarchy

We are currently working on using our hierarchical semi-supervised learning technique, GLOFIN (refer Chapter 6), for the task of word sense disambiguation with respect to WordNet. Here we consider occurrences of all unambiguous words as training data and ambiguous word occurrences as unlabeled data. In these experiments, we use the WordNet synset hierarchy to deduce ontological constraints. Figure 8.2 shows an example of synset hierarchy around word 'goal'. Further we plan to use clues from two different data views: glosses and example usages of the word senses.

Exploratory Relation Extraction

Most techniques discussed in this thesis are not tied to any particular document representation, choice of features or task. Hence one obvious future research direction lies in applying these ideas to tasks other than concept extraction. For example, Relation extraction is a very challenging task, and many ontologies like NELL that are rich in terms of coverage of concepts have low coverage in terms of relations between entities. We hope that the Exploratory learning technique can be extended for the relation extraction task.

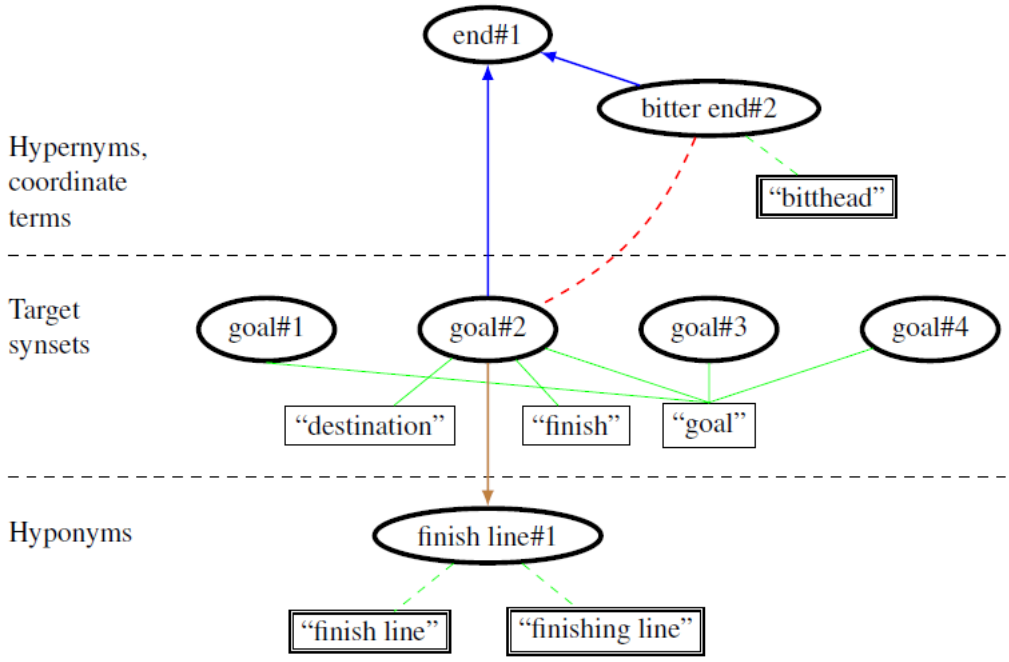


Figure 8.2: Example synset hierarchy for a word "goal". There are four target synsets 'goal#1' to 'goal#4'. 'finish line#1' is a hyponym of goal#2 and 'end#1' is a hypernym. 'goal#2' and 'bitter end#2' are mutually exclusive. 'finish line' is unambiguous and 'destination' is ambiguous.

Appendix A

WebSets: Table Driven Information Extraction

In this Chapter we cover some additional details about our WebSets project that are not covered in depth in Chapter 3.

A.1 Our Method

A.1.1 Table Identification

Currently WebSets parses tables defined by <table> tags[17]. This is only a fraction of structured data available on the Web. Use of other techniques like Gatterbauer et al. [51] can provide more input data to learn sets from.

Further only a small fraction of HTML tables actually contain useful relational data(see Section A.2). Remaining tables are used for formatting or rendering purposes rather than to present relational data. To filter out useful tables, WebSets uses the following set of features: (1) the number of rows (2) the number of non-link columns (3) the length of cells after removing formatting tags (4) whether table contains other HTML tables. The thresholds set for our experiments are explained in Section A.2.

A.1.2 Entity Clustering

At the end of the table identification step, we have a collection of HTML tables which are likely to contain relational data. Each of the table-cells is a candidate entity and each table-column is a candidate set of entities. However many columns have information that is useful only within a site (e.g., navigational links) and many are overlapping. To solve this problem we use the redundancy of information on the Web. If a set of entities co-occur in multiple table-columns across the Web, then it is more likely to be an important entity set. Since these tables come from different domains and are created by different authors, they will typically only partially

[17]We found that large number of HTML pages have broken syntax. We use the HTML syntax cleaning software Tidy [1] to fix the syntax in these pages.

Country	Capital City
India	Delhi
China	Beijing
Canada	Ottawa
France	Paris

Table A.1: TableId= 21, domain= "www.dom1.com"

overlap. To cluster the entities in these columns into coherent sets, the first essential step will be to represent this data in a way that reveals the co-occurrence of entities across multiple table columns. In this section, we will first discuss the data representation, and then algorithms for entity clustering.

Data Representation

We considered two ways to represent table data. With the "Entity record representation", one record per entity is created, and it contains information about which table-columns and URL domains the entity was mentioned in. These entity records can then be clustered, based on the overlap of table-columns and domains, to yield sets of entities. One advantage of this representation is that it is compact. Another advantage is that the number of distinct domains an entity occurred in can be used as an indicator of how "important" it is. A disadvantage of this representation is that if some entity-name is ambiguous (has multiple senses), it will collapse all senses of the entity into one record. For example, consider a corpus which contains mentions of the entity "Apple" in two senses, "Apple as a fruit" and "Apple as a company". This representation will create a single record for the entity "Apple" with its occurrence as a fruit and as a company confounded. An unsupervised IE system might find it difficult to decompose multiple senses from this single record.

To solve this problem, we propose a novel representation of the table data called "Entity-Triplet records". In this representation, there is a record for each triplet of adjacent entities instead of for individual entities. Each record contains information about which table-columns and URL domains the triplet occurred in. Hence in case of an ambiguous entity like "Apple", its occurrences as a fruit will be separate from its occurrences as a company, e.g., {Apple, Avocado, Banana} will be one triplet record and {Apple, Microsoft, DELL} will be another triplet record. Thus entities within a triplet disambiguate each other. Since we have a list of all domains a triplet occurred in, only those triplets which appear in enough domains can be considered as important.

In this way, we represent the table data as a triplet store, which contains a record for each entity triplet. The triplet store can be built in one single pass over the corpus. If the system considers all possible triplets that can be created from each table column, then number of triplet records will be quadratic in the total size of all tables. This can be a serious concern for a web-scale dataset. Hence our system constructs only those triplets which are adjacent i.e. subsequences of entities in a table-column. This ensures that number of triplet records are linear in total size of all tables, making the system scalable. The system keeps track of all table-columns a triplet occurred in, which makes it possible to reconstruct a column by joining triplets on columnId. Hence this storage method does not result in any loss of information.

Consider an example of tables containing countries and their capitals. Original tables are

Country	Capital City
China	Beijing
Canada	Ottawa
France	Paris
England	London

Table A.2: TableId= 34, URL= "www.dom2.com"

Entities	Tid:Cids	Domains
India,China,Canada	21:1	www.dom1.com
China, Canada, France	21:1, 34:1	www.dom1.com, www.dom2.com
Delhi, Beijing, Ottawa	21:2	www.dom1.com
Beijing, Ottawa, Paris	21:2, 34:2	www.dom1.com, www.dom2.com
Canada, England, France	34:1	www.dom2.com
London, Ottawa, Paris	34:2	www.dom2.com

Table A.3: Triplet records created by WebSets

shown in Table A.1, A.2 and the triplet records created by WebSets are shown in Table A.3. Second row in Table A.3, indicates that the triplet (China, Canada, France) occurred in column 1 of tableId 21 and column 1 of tableId 34. Also these entities are retrieved from webpages which reside in the domains "www.dom1.com" and "www.dom2.com".

These triplets are canonicalized by converting entity strings to lower case and arranging the constituent entities in alphabetical order. The triplets are then ranked in descending order of number of domains.

We create $O(n)$ triplets from a table column of size n. Adding each triplet to the Triplet Store using hashmap takes $O(1)$ time. Given a set of T HTML tables with a total of N entities in them, the Triplet Store can be created in $O(N)$ time. Ranking the triplets using any common sorting technique will take $O(N * logN)$ time. Hence the total complexity of building the Triplet Store is $O(N * logN)$.

Building entity clusters

The next task is to cluster these triplet records into meaningful sets. The system does not know how many clusters are present in the underlying dataset; and since our dataset will be constructed from a huge HTML web corpus, the clustering algorithm needs to be very efficient. Given these requirements, it can be easily seen that parametric clustering algorithms like K-means may not be effective due to the unknown number of clusters. Non-parametric algorithms like agglomerative clustering [35] fit most of our requirements. The most efficient agglomerative clustering algorithm is the single-link clustering algorithm, but even that would require computing the similarity of every pair of entity triplets returned. This will be very expensive for the web scale datasets we are aiming to handle. Hence we develop a new bottom-up clustering algorithm which is efficient in terms of both space and time. Experiments to compare our algorithm with a standard K-means clustering algorithm are described in Section A.2.

Computational complexity

Suppose that our dataset has total T tables. Let these tables have in total N cells. For each triplet t, Algorithm 1 finds entity and column overlap with all existing clusters. This operation can be implemented efficiently by keeping two inverted indices: (1) from each entity to all clusterIds it belongs to and (2) from each columnId to all clusterIds it belongs to. ClusterIds in each "postings list "[18] will be kept in sorted order.

Merging k sorted lists, with resultant list size of n takes $O(n * logk)$ time. Now let us compute worst case time complexity of Algorithm 1. To compute entity overlap of each triplet, the algorithm merges 3 postings lists with total size of $O(N)$. This step will take $O(N)$ time. To compute TableId:ColumnId overlap of each triplet, it merges $O(T)$ postings lists with total size of $O(N)$. This step will take $O(N * logT)$ time. Hence for each triplet, finding a clusterId to merge the triplet with takes $O(N * logT)$ time. There are $O(N)$ triplets. So the ***Bottom-Up Clusterer*** will have worst case time complexity of $O(N^2 * logT)$. In practice, it is much less than this. If N is total number of table-cells in the corpus then the total number of triplet occurrences is also $O(N)$. Hence all the postings list merges can be amortized to be $O(N)$. Hence amortized complexity of this clustering algorithm is $O(N * logT)$. Considering the time complexity to sort the triplet store, total time complexity is $O(N * logN)$, which is much better than the complexity of a naive single-link clustering algorithm, which is $O(N^2 * logN)$.

A.1.3 Hypernym Recommendation

Previous sections described how do we obtain coordinate term clusters using co-occurrence of terms in the table columns. In this section we label these term clusters with the help of Hyponym Concept dataset.

Building The Hyponym-Concept Dataset

The *Hyponym Concept Dataset* is built by acquiring concept-instance pairs from unstructured text using Hearst patterns. For this task we used the data extracted from the ClueWeb09 corpus [23] by the developers of the NELL KB [91]. They used heuristics to identify and then shallow-parse approximately 2 billion sentences, and then extracted from this all patterns of the form "_ $word_1$ .. $word_k$ _" where the 'filler' $word_1$ .. $word_k$ is between one and five tokens long, and this filler appears at least once between two base noun phrases in the corpus. Each filler is paired with all pairs of noun phrases that bracket it, together with the count of the total number of times this sequence occurred. For instance, the filler '_ and vacations in _' occurs with the pair 'Holidays, Thailand' with a count of one and 'Hotels,Italy' with a count of six, indicating that the phrase "Hotels and vacations in Italy" occurred six times in the corpus . The hyponym dataset was constructed by finding all fillers that match one of the regular expressions in Table A.4. These correspond to a subset of the Hearst patterns used in ASIA [136] together with some "doubly anchored" versions of these patterns [73].

[18]In the information retrieval terminology, an inverted index has a record per word that contains the list of all document-ids the word occurred in. This list is referred to as the "postings list". In this paper, a "postings list" refers to the list of all clusterIds that a triplet or a columnId belongs to.

Id	Regular expression
1	arg1 such as (w+ (and|or))? arg2
2	arg1 (w+)? (and|or) other arg2
3	arg1 include (w+ (and|or))? arg2
4	arg1 including (w+ (and|or))? arg2

Table A.4: Regular expressions used to create Hyponym Concept Dataset

Hyponym	Concepts
USA	country:1000
India	country:200
Paris	city:100, tourist_place:50
Monkey	animal:100, mammal:60
Sparrow	bird:33

Table A.5: An example of Hyponym Concept Dataset

Each record in the Hyponym Concept Dataset contains an entity and all concepts it co-occurred with. Table A.5 shows an example of records in this dataset. According to this table, entity "USA" appeared with the concept "country" 1000 times. Similarly, "Monkey" appeared with two different concepts, 100 times with "animal" and 60 times with "mammal".

Assigning Hypernyms to Clusters

Assigning a meaningful concept-name to each entity set is important for two reasons. It enables us to systematically evaluate entity sets; e.g., it is easier for an evaluator to answer the question: "Is Boston a city?" than "Is Boston a member of set #37?") It also makes the system more useful for summarizing the data in a corpus (see Section 3.1.3). This section describes how WebSets recommends candidate hypernyms for each entity set produced by Algorithm 1.

For this task, we use the coordinate term clusters extracted from tables and *Hyponym Concept Dataset* extracted using Hearst patterns. Note that this part of the system uses information extracted from unstructured text from the Web, to recommend category names to sets extracted from tables on the Web.

A.2 Evaluation of Individual Stages of WebSets

In this section we evaluate each stage of WebSets system and measure the performance. Here we consider WebSets as a technique to generate large number of concept-instance pairs given a HTML corpus.

A.2.1 Evaluation: Table Identification

Table A.6 shows the statistics of table identification for each dataset. Based on the features described in Section A.1.1, we filter out only those tables as useful which cross some predefined thresholds. These thresholds were derived from the intuitions after manually going through some samples of data, and are kept constant for all datasets and experiments described in this paper.

Dataset	#Tables	%Rela--tional	#Filtered tables	%Rela--tional filtered	#Triplets
Toy_Apple	2.6K	50	762	75	15K
Delicious_Sports	146.3K	15	57.0K	55	63K
Delicious_Music	643.3K	20	201.8K	75	93K
CSEAL_Useful	322.8K	30	116.2K	80	1148K
ASIA_NELL	676.9K	20	233.0K	55	421K
ASIA_INT	621.3K	15	216.0K	60	374K
Clueweb_HPR	586.9K	10	176.0K	35	78K

Table A.6: Table Identification Statistics

When we evaluated a random sample of recursive tables, 93% of them were useless for our purposes, and only 7% tables contained relational data. Hence we decide to ignore the recursive tables. We construct triplets of entities in a table column, hence a table should have at least 3 rows. As we are not using any link data in our system, we consider only those columns which do not have links. WebSets is looking for tables containing relational data, hence if a table has at least 2 non-link columns, probability of it having relational data increases. While counting these rows and columns, we are looking for named entities. So from each cell, all HTML tags and links are removed. A very coarse filtering by length is then applied. We consider only those table cells which are 2-50 characters in length. Table A.6 shows that percentage of relational tables increases by orders of magnitude due to this filtering step.

A.2.2 Evaluation: Clustering Algorithm

In these set of experiments we first compare WebSets with baseline K-means clustering algorithm. Later we see how "Entity-Triplet record" representation is better than "Entity record" representation. We compare the clustering algorithms in terms of commonly used clustering metrics: cluster purity (Purity), normalized mutual information (NMI), rand index (RI) [82] and Fowlkes-Mallows index (FM) which are defined as follows:

- **Cluster Purity:** To compute cluster purity, for each cluster the class which is most frequent in it gets assigned. The accuracy of this assignment is then measured by counting the number of correctly assigned documents and dividing by total number of documents.
  $purity(\Omega, C) = \frac{1}{N}\sum_k \max_j |\omega_k \bigcap c_j|$ where
  N is total number of documents, $\Omega = \omega_1, \omega_2, ..., \omega_K$
  is the set of clusters and $C = c_1, c_2, ..., c_J$ is the set of classes. We interpret ω_k as the set of documents in cluster ω_k and c_j as the set of documents in cluster c_j.
- **Normalized Mutual Information:** High purity is easy to achieve when the number of clusters is large, hence purity cannot be used to trade off the quality of the clustering against the number of clusters. NMI is a measure that allows us to make such tradeoff.
  $NMI(\Omega, C) = \frac{I(\Omega;C)}{[H(\Omega)+H(C)]/2}$
  where maximum likelihood estimates for mutual information and entropy are computed as follows:
  $I(\Omega, C) = \sum_k \sum_j \frac{|\omega_k \bigcap c_j|}{N}$, $H(\Omega) = -\sum_k \frac{|\omega_k|}{N} * log\frac{|\omega_k|}{N}$.

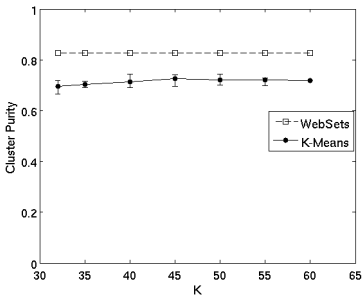
- **Rand Index:** This metric is based on information-theoretic interpretation of clustering. Clustering is a series of decisions, one for each of the $N(N-1)/2$ pairs of documents in the collection. True label denotes the pair belongs to same class or not. Predicted label denotes whether the pair belongs to same cluster or not. This way we can count True Positive(TP), False positive (FP), True Negative (TN) and False Negative (FN) score for the clustering. Rand Index is then defined as follows: $RI = \frac{TP+TN}{TP+FP+FN+TN}$.
- **Fowlkes-Mallows index:** While the previous three metrics are applicable only for hard-clustering, this metric is defined for soft clustering of data where clusters can be overlapping but classes are non-overlapping. Let n_{ij} be the size of intersection of cluster ω_i and class c_j, $n_{i*} = \sum_j n_{ij}$ and $n_{*j} = \sum_i n_{ij}$.
  $FM = (\sum_{ij} \binom{n_{ij}}{2}) / \sqrt{\sum_i \binom{n_{i*}}{2} \sum_j \binom{n_{*j}}{2}}$
  The derivation of this formula can be found in Ramirez et al. [106].

Here each triplet is considered as a document. Ω refers to clusters of these triplets generated by WebSets or K-means. C refers to actual classes these triplets belong to. To compare quality of clusters across algorithms, we manually labeled each table-column of Toy_Apple and Delicious_Sports datasets. These labels are then extended to triplets within the table column. This experiment is not repeated for remaining datasets because manual labeling of the whole dataset is very expensive.

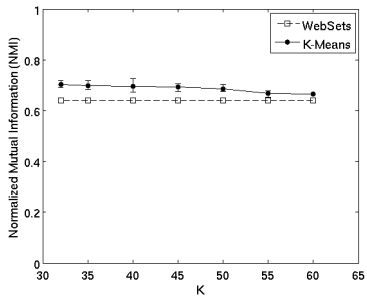
The performance of K-means depends on the input parameter K and random initialization of cluster centroids to start the clustering process. We run K-means with cosine distance function, a range of values of K and multiple starting points for each value of K. Figure A.1 shows the plot of various runs of K-means vs. WebSets on Delicious_Sports dataset. Table A.7 shows the comparison of WebSets vs. best run of K-means on Toy_Apple and Delicious_Sports datasets. We can see that WebSets performs better or comparable to K-means in terms of purity, NMI, RI, and FM. Through manual labeling we found that there are 27 and 29 distinct category sets in Toy_Apple and Delicious_Sports datasets respectively. We can see that WebSets defined 25 and 32 distinct clusters which are very close to actual number of meaningful sets, compared to 40 and 50 clusters defined by K-means.

Standard K-means algorithm has time complexity of $O(I * K * N * T)$, where I is the number of iterations, K is the number of clusters, N is the number of table-cells and T is the number of dimensions (here number of table-columns). WebSets has time complexity of $O(N * logN)$. Hence our bottom-up clustering algorithm is more efficient than K-means in terms of time-complexity.

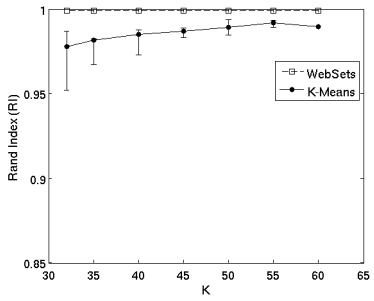
To study entity disambiguation effect of triplet records, we generated both "Entity record" and "Entity-Triplet record" representations of Toy_Apple dataset. When we ran our clustering algorithm on "Entity-Triplet record" dataset, we found "Apple" in two different clusters, one in which it was clustered with other fruits and had supporting evidence from table-columns talking about fruits; other cluster contained companies and evidence from related table-columns. Running the same clustering algorithm on "Entity record" dataset, resulted in a huge cluster containing "Apple", fruits and companies all combined. Thus we can say that entity triplets do help WebSets to disambiguate multiple senses of the same entity-string.



(a) Cluster Purity



(b) Normalized Mutual Information



(c) Rand Index

Figure A.1: Comparison of WebSets and K-Means algorithms on Delicious_Sports

Dataset	Method	K	Purity	NMI	RI	FM
Toy_Apple	K-means	40	0.96	0.71	0.98	0.41
	WebSets	25	0.99	0.99	1.00	0.99
Delicious_Sports	K-means	50	0.72	0.68	0.98	0.47
	WebSets	32	0.83	0.64	1.00	0.85

Table A.7: Comparison of WebSets vs. K-means

Now we compare the performance of clustering algorithms on entity record vs. triplet record representation. In this experiment we use Toy_Apple dataset. A table-column is considered as document and clusters produced are soft-clustering of this document set. Hence each table-column can be present in multiple clusters, but belongs to only one class. Purity, NMI and RI metrics are not applicable for soft clustering, however FM metric is valid. We run K-Means algorithm for different values of K. Table A.8 shows the best performing results of K-Means. WebSets produced 25 clusters on entity record representation and 34 clusters using triplet representation. In terms of FM index, each method gives better performance on triplet record representation when compared to entity record representation. Hence triplet record representation does improve these clustering methods.

Method	K	FM w/ Entity records	FM w/ Triplet records
WebSets		0.11 (K=25)	0.85 (K=34)
K-Means	30	0.09	0.35
	25	0.08	0.38

Table A.8: Comparison of performance on Entity vs. Triplet record representation (Toy_Apple dataset)

A.2.3 Evaluation: Hyponym-Concept Dataset

Note that the Hyponym-concept dataset is created in an unsupervised way, hence we cannot guarantee that the concept-instance pairs in this dataset are accurate. To get an idea of quality of concept-instance pairs in this dataset, we randomly sampled 100 pairs. 55% of them were accurate. Hence hypernym recommendation step is dealing with noisy concept-instance pairs as input.

A.2.4 Evaluation: Hypernym Recommendation

We compare our hypernym recommendation technique with Van Durme and Pasca technique [132]. Our method is different from Van Durme and Pasca Method (DPM) in the sense that, they output a concept-instance pair only when it appears in a set of candidate concept-instance pairs i.e. it exists in Hyponym-concept dataset. In our method, based on overlap of coordinate term cluster with the Hyponym Concept dataset, we extend the labels to whole cluster. Another difference is the coordinate term clusters we are dealing with. DPM assumes term clusters are semantic partitions of terms present in the text corpus. The clusters generated by WebSets are clusters of table columns. There is higher possibility of having multiple small size clusters which all belong to same semantic class, but were not merged due to our single pass bottom-up clusterer. Hence the same method may not work equally well on these clusters.

We sample 100 concept-instance pairs randomly from output of each method to measure accuracy. The results of different methods are presented in Table A.9. As we can see, DPM generates concept-instance pairs with 50% accuracy even when run with conservative thresholds like $K = 5$ and $J = 0.2$. We also tried an extension of DPM called DPMExt which outputs a label for each entity in the cluster, when the label satisfies thresholds defined by J and K. This extension increases coverage of concept-instance pairs orders of magnitude (0.4K to 1.2K) at the cost of slight decrease in accuracy (50% to 44%). Hypernym recommendation of WebSets (WS) is described in Algorithm 2, and only topmost hypernym in the ranked list is produced for each cluster. Table A.9 shows that WS has a reasonable accuracy (62.2%) and yield compared to DPM and DPMExt. As discussed in Section A.2.3, Hyponym-concept dataset has noisy pairs. We also tried an extension of WebSets called WSExt which overcomes this problem by considering only those class-instance pairs which have occurred at least 5 times in the corpus. Adding this simple constraint, improves accuracy from 67% to 78% and correct pairs yield from 45K to 51K.

Method	K	J	%Accuracy	yield (#pairs produced)	#correct pairs (predicted)
DPM	inf	0.0	34.6	88.6K	30.7K
DPM	5	0.2	50.0	0.8K	0.4K
DPMExt	inf	0	21.9	100,828.0K	22,081.3K
DPMExt	5	0.2	44.0	2.8K	1.2K
WS	-	-	67.7	73.7K	45.8K
WSExt	-	-	78.8	64.8K	51.1K

Table A.9: Comparison of various methods in terms of accuracy and yield on CSEAL_Useful dataset

Bibliography

[1] Html TIDY project. `http://tidy.sourceforge.net/`. 5, 17

[2] Freebase. *http://freebase.com*. 7.6

[3] Marco D Adelfio and Hanan Samet. Schema extraction for tabular data on the web. *Proceedings of the VLDB Endowment*, 6(6):421–432, 2013. 3.2

[4] Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. Random walks for knowledge-based word sense disambiguation. *Comput. Linguist.*, 40(1):57–84, March 2014. ISSN 0891-2017. doi: 10.1162/COLI_a_00164. 6.4.2

[5] AIMMS. The MOSEK toolkit. 7.3.2

[6] Ramnath Balasubramanyan, **Bhavana Dalvi**, and William W Cohen. From topic models to semi-supervised learning: Biasing mixed-membership models to exploit topic-indicative features in entity clustering. In *Machine Learning and Knowledge Discovery in Databases*, pages 628–642. Springer, 2013. 3.2

[7] Arindam Banerjee, Inderjit S Dhillon, Joydeep Ghosh, and Suvrit Sra. Clustering on the unit hypersphere using von Mises-Fisher distributions. In *Journal of Machine Learning Research*, pages 1345–1382, 2005. 4.1.3

[8] Satanjeev Banerjee and Ted Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 805–810, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc. 6.4.2

[9] Sugato Basu, Arindam Banerjee, and Raymond Mooney. Semi-supervised clustering by seeding. In *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*, 2002. 4.1.3

[10] Steffen Bickel and Tobias Scheffer. Multi-view clustering. In *International Conference on Data Mining*, volume 4, pages 19–26, 2004. 1.1.2, 1.1.4, 5.1, 5.2.3, 5.3.2, 5.5, 5.6, 8.1

[11] David M. Blei, Thomas L Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum. Hierarchical topic models and the nested Chinese restaurant process. volume 16, page 17. The MIT Press, 2004. 4.5, 7.6

[12] David M Blei, Thomas L Griffiths, and Michael I Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. volume 57, page 7. ACM, 2010. 7.6

[13] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training.

In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 92–100. ACM, 1998. 5.1, 5.5

[14] Charles Bouveyron. Adaptive mixture discriminant analysis for supervised learning with unobserved classes. *Journal of Classification*, 31(1):49–84, 2014. 4.6

[15] Ulf Brefeld and Tobias Scheffer. Semi-supervised learning for structured output variables. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 145–152, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143863. 5.5

[16] Ulf Brefeld, Christoph Büscher, and Tobias Scheffer. Multi-view discriminative sequential learning. In *Machine Learning: European Conference on Machine Learning and Knowledge Discovery in Databases 2005*, pages 60–71. Springer, 2005. 5.5

[17] Ulf Brefeld, Thomas Gärtner, Tobias Scheffer, and Stefan Wrobel. Efficient co-regularised least squares regression. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 137–144, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143862. 5.5

[18] Kenneth P Burnham and David R Anderson. Multimodel inference understanding AIC and BIC in model selection. *Sociological methods & research*, 33(2):261–304, 2004. 4.1.2

[19] Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. Webtables: Exploring the power of tables on the Web. *Proceedings of the VLDB Endowment*, 1 (1):538–549, 2008. 3.2

[20] Deng Cai, Xuanhui Wang, and Xiaofei He. Probabilistic dyadic data analysis with local and global consistency. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 105–112. ACM, 2009. 4.2

[21] Lijuan Cai and Thomas Hofmann. Hierarchical document categorization with support vector machines. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, pages 78–87. ACM, 2004. 7.6

[22] Xiao Cai, Feiping Nie, and Heng Huang. Multi-view K-means clustering on big data. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 2598–2604. AAAI Press, 2013. ISBN 978-1-57735-633-2. 5.5

[23] Jamie Callan. The ClueWeb09 dataset. `http://boston.lti.cs.cmu.edu/Data/clueweb09/`. 7, 5.3.1, 7.4.1, A.1.3

[24] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *In AAAI Conference on Artificial Intelligence*, 2010. 2.3, 2.1(a), 6.3.3

[25] Andrew Carlson, Justin Betteridge, Richard C Wang, Estevam R Hruschka Jr, and Tom M Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, pages 101–110. ACM, 2010. (document), 1.3, 2.3, 3.1.2, 3.2, 3.3.2, 3.3.2, 4.6, 5.1, 5.1, 7, 7.1, 7.6

[26] David Carmel, Ming-Wei Chang, Evgeniy Gabrilovich, Bo-June (Paul) Hsu, and Kuansan

Wang. ERD 2014: Entity recognition and disambiguation challenge. 48(2):63–77, 2014. 1.3, 6.3.1, 6.3.3

[27] Carlos Castillo, Héctor Valero, José Guadalupe Ramos, and Josep Silva. Information extraction from webpages based on DOM distances. In *Computational Linguistics and Intelligent Text Processing*, pages 181–193. Springer, 2012. 3.2

[28] Gilles Celeux and Gérard Govaert. A classification EM algorithm for clustering and two stochastic versions. *Computational Statistics & Data Analysis*, 14(3):315–332, 1992. 5.2.1, 5.2.2

[29] Ming-Wei Chang, Lev Ratinov, and Dan Roth. Guiding semi-supervision with constraint-driven learning. In *Annual Meeting-Association for Computational Linguistics*, volume 45, page 280, 2007. 6.4.1

[30] Mark Ming-Tso Chiang and Boris Mirkin. Intelligent choice of the number of clusters in K-Means clustering: An experimental study with different cluster spreads. *Journal of Classification*, 27(1):3–40, 2010. 4.6

[31] Francis CY Chik, Robert WP Luk, and Korris FL Chung. Text categorization based on subtopic clusters. In *Natural Language Processing and Information Systems*, pages 203–214. Springer, 2005. 6.2.2

[32] Namyoun Choi, Il-Yeol Song, and Hyoil Han. A survey on ontology mapping. *ACM SIGMOD Record*, 35(3):34–41, September 2006. ISSN 0163-5808. doi: 10.1145/1168092.1168097. 6.4.2

[33] C. Mario Christoudias, Raquel Urtasun, and Trevor Darrell. Multi-View Learning in the Presence of View Disagreement. In *Uncertainty in Artificial Intelligence*, 2012. 5.4.2

[34] Jeffrey Dalton, Laura Dietz, and James Allan. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 365–374, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2257-7. doi: 10.1145/2600428.2609628. 6.3.1, 6.4.2

[35] William HE Day and Herbert Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. volume 1, pages 7–24. Springer, 1984. A.1.2

[36] Flavio De Benedictis, Stefano Faralli, and Roberto Navigli. Glossboot: Bootstrapping multilingual domain glossaries from the web. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 528–538, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. 6.4.2

[37] Inderjit S Dhillon and Dharmendra S Modha. Concept decompositions for large sparse text data using clustering. *Machine learning*, 42(1-2):143–175, 2001. 5.2, 5.2.1, 5.2.2

[38] Tom Diethe, DavidRoi Hardoon, and John Shawe-Taylor. Constructing nonlinear discriminants from multiple data views. In JosLuis Balczar, Francesco Bonchi, Aristides Gionis, and Michle Sebag, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 6321 of *Lecture Notes in Computer Science*, pages 328–343. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15879-7. doi: 10.1007/978-3-642-15880-3_27. 5.5

[39] Weisi Duan and Alexander Yates. Extracting glosses to disambiguate word senses. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 627–635, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. ISBN 1-932432-65-5. 6.4.2

[40] Haimonti Dutta, Rebecca J Passonneau, Austin Lee, Axinia Radeva, Boyi Xie, and David Waltz. Learning parameters of the K-means algorithm from subjective human annotation. In *Twenty-Fourth International FLAIRS Conference*, 2011. 4.6

[41] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. Web-scale information extraction in knowitall:(preliminary results). In *Proceedings of the 13th International Conference on World Wide Web*, pages 100–110. ACM, 2004. 3.2, 4.6

[42] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. volume 165, pages 91–134. Elsevier, 2005. 3.2

[43] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 2008. 6.3.3

[44] Stefano Faralli and Roberto Navigli. A new minimally-supervised framework for domain word sense disambiguation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1411–1422, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. 6.4.2

[45] Jason Farquhar, David Hardoon, Hongying Meng, John S Shawe-taylor, and Sandor Szedmak. Two view learning: Svm-2k, theory and practice. In *Advances in neural information processing systems*, pages 355–362, 2005. 5.5

[46] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998. 6.4.2

[47] Nir Friedman, Matan Ninio, Itsik Pe'er, and Tal Pupko. A structural em algorithm for phylogenetic inference. *Journal of Computational Biology*, 9(2):331–353, 2002. 4.1.1

[48] Masatoshi Fujii, Saichi Izumino, Ritsuo Nakamoto, and Yuki Seo. Operator inequalities related to Cauchy-Schwarz And Holder-McCarthy inequalities. In *Nihonkai Mathematical Journal*, 1997. 5.2.1

[49] Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 11:2001–2049, 2010. 6.4.1

[50] Kuzman Ganchev, Joao Graca, John Blitzer, and Ben Taskar. Multi-view learning over structured and non-identical outputs. *arXiv preprint arXiv:1206.3256*, 2012. 6.4.1

[51] Wolfgang Gatterbauer, Paul Bohunsky, Marcus Herzog, Bernhard Krüpl, and Bernhard Pollak. Towards domain-independent information extraction from web tables. In *Proceedings of the 16th International Conference on World Wide Web*, pages 71–80. ACM,

2007. 3.1.2, 3.2, A.1.1

[52] Lisa Getoor. Linqs datasets `http://linqs.cs.umd.edu/projects//projects/lbc/index.html`. 5.3.1

[53] Zoubin Ghahramani, Michael I Jordan, and Ryan P Adams. Tree-structured stick breaking for hierarchical data. In *Advances in Neural Information Processing Systems*, pages 19–27, 2010. 7.6

[54] Sean Gilpin, Siegfried Nijssen, and Ian Davidson. Formalizing hierarchical clustering as integer linear programming. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, pages 372–378, 2013. 5.1, 5.5

[55] Siddharth Gopal, Yiming Yang, Bing Bai, and Alexandru Niculescu-Mizil. Bayesian models for large-scale hierarchical classification. In *Advances in Neural Information Processing Systems*, pages 2411–2419, 2012. 7.6

[56] Rahul Gupta and Sunita Sarawagi. Answering table augmentation queries from unstructured lists on the web. volume 2, pages 289–300. VLDB Endowment, 2009. 3.2

[57] Rahul Gupta and Sunita Sarawagi. Joint training for open-domain extraction on the web: exploiting overlap when supervision is limited. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, pages 217–226. ACM, 2011. 3.2

[58] Hannaneh Hajishirzi, Leila Zilles, S. Daniel Weld, and Luke Zettlemoyer. Joint coreference resolution and named-entity linking with multi-pass sieves. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 289–299. Association for Computational Linguistics, 2013. 6.4.2

[59] Greg Hamerly and Charles Elkan. Learning the K in K-Means. In *Advances in Neural Information Processing Systems*, 2003. 4.6

[60] Xianpei Han, Le Sun, and Jun Zhao. Collective entity linking in web text: A graph-based method. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 765–774, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0757-4. doi: 10.1145/2009916.2010019. 6.4.2

[61] Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics, 1992. 3, 5

[62] Fangwei Hu, Tianqi Chen, Nathan N. Liu, Qiang Yang, and Yong Yu. Discriminative factor alignment across heterogeneous feature space. In *Proceedings of the 2012 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part II*, ECML PKDD'12, pages 757–772, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-33485-6. doi: 10.1007/978-3-642-33486-3_48. 5.5

[63] David Insa, Josep Silva, and Salvador Tamarit. Using the words/leafs ratio in the dom tree for content extraction. *The Journal of Logic and Algebraic Programming*, 82(8):311–325, 2013. 3.2

[64] Heng Ji and Ralph Grishman. Knowledge base population: Successful approaches and

challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1148–1158. Association for Computational Linguistics, 2011. 2.2

[65] Heng Ji, Ralph Grishman, and Hoa Trang Dang. Overview of the TAC 2011 knowledge base population track. In *Text Analysis Conference*, 2011. 6.3.1

[66] Xin Jin, Fuzhen Zhuang, Shuhui Wang, Qing He, and Zhongzhi Shi. Shared structure learning for multiple tasks with multiple views. In *Machine Learning and Knowledge Discovery in Databases*, volume 8189 of *Lecture Notes in Computer Science*, pages 353–368. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-40990-5. doi: 10.1007/978-3-642-40991-2_23. 5.5

[67] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967. 5.1

[68] Jaap Kamps, Rianne Kaptein, and Marijn Koolen. Using anchor text, spam filtering and wikipedia for web search and entity ranking. *Text REtrieval Conference*, 2010. 7

[69] Yoonseop Kang and Seungjin Choi. Restricted deep belief networks for multi-view learning. In *Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part II*, ECML PKDD'11, pages 130–145, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-23782-9. 5.5

[70] Shiva Prasad Kasiviswanathan, Prem Melville, Arindam Banerjee, and Vikas Sindhwani. Emerging topic detection using dictionary learning. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 745–754. ACM, 2011. 4.6

[71] Adam Kilgarriff and Joseph Rosenzweig. English SENSEVAL: Report and Results. In *Language Resources and Evaluation*, 2000. 6.4.2

[72] Sokol Koço and Cécile Capponi. A boosting approach to multiview classification with cooperation. In *Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part II*, ECML PKDD'11, pages 209–228, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-23782-9. 5.5

[73] Zornitsa Kozareva and Eduard Hovy. A semi-supervised method to learn and construct taxonomies using the web. In *Proceedings of the Conference on empirical methods in natural language processing*, 2010. A.1.3

[74] Jayant Krishnamurthy and Tom M. Mitchell. Which noun phrases denote which concepts? In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 570–580, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-87-9. 2.2, 6.3.3

[75] Wen-Yu Lee, Liang-Chi Hsieh, Guan-Long Wu, and Winston Hsu. Graph-based semi-supervised learning with multi-modality propagation for large-scale image datasets. *Journal of Visual Communication and Image Representation*, 24(3):295–302, 2013. 6.4.1

[76] Verayuth Lertnattee and Thanaruk Theeramunkong. Effect of term distributions on

centroid-based text categorization. *Information Sciences*, 158:89–115, 2004. 4.1.3

[77] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, SIGDOC '86, pages 24–26, New York, NY, USA, 1986. ACM. ISBN 0-89791-224-1. doi: 10.1145/318723.318728. 6.3.1, 6.4.2

[78] Boaz Leskes and Leen Torenvliet. The value of agreement a new boosting algorithm. *18th Annual Conference on Learning Theory*, pages 557–586, 2005. ISSN 0022-0000. doi: 10.1016/j.jcss.2007.06.005. 5.5

[79] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. Annotating and searching web tables using entities, types and relationships. *Proceedings of the VLDB Endowment*, 3 (1-2):1338–1347, 2010. 3.2

[80] Thomas Lin, Mausam, and Oren Etzioni. Entity linking at web scale. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12, pages 84–88, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. 6.4.2

[81] Charles Mallah, James Cope, and James Orwell. Plant leaf classification using probabilistic integration of shape, texture and margin features. *Signal Processing, Pattern Recognition and Applications*, 2013. 5.3.1

[82] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. Introduction to information retrieval. In *Cambridge University Press*, 2008. 3.1.2, 7.6, A.2.2

[83] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *ACL demo*, 2014. 6.3.2

[84] David Martinez, Oier Lopez de Lacalle, and Eneko Agirre. On the use of automatically acquired examples for all-nouns word sense disambiguation. *J. Artif. Int. Res.*, 33(1): 79–107, September 2008. ISSN 1076-9757. 6.3.3

[85] Mohammad M Masud, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani Thuraisingham. Integrating novel class detection with classification for concept-drifting data streams. In *Machine Learning and Knowledge Discovery in Databases*, pages 79–94. Springer, 2009. 4.6

[86] Kathryn Mazaitis, Richard C. Wang, Frank Lin, **Bhavana Dalvi**, Jakob Bauer, and William W. Cohen. A tale of two entity linking and discovery systems in KBP-TAC 2014. In *KBP Entity Linking Task*, 2014. 1.3

[87] Tara McIntosh. Unsupervised discovery of negative categories in lexicon bootstrapping. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 356–365. Association for Computational Linguistics, 2010. 4.6

[88] Daniel A Menascé, Virgilio AF Almeida, Rodrigo Fonseca, and Marco A Mendes. A methodology for workload characterization of e-commerce sites. In *Proceedings of the 1st ACM Conference on Electronic commerce*, pages 119–128. ACM, 1999. 4.6

[89] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed

representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013. 7.6

[90] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 1995. 6.3.1

[91] Tom Mitchell. Nell: Never-ending language learning. *http://rtw.ml.cmu.edu/rtw/*. 3.2, 6.3.1, 6.3.3, A.1.3

[92] Thahir P. Mohamed, Estevam R. Hruschka, Jr., and Tom M. Mitchell. Discovering relations between noun categories. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1447–1455, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-937284-11-4. 1.3, 4.6, 7.6

[93] Andrea Moro, Alessandro Raganato, and Roberto Navigli. Entity linking meets word sense disambiguation: a unified approach. *TACL*, 2:231–244, 2014. 6.4.2

[94] Dana Movshovitz-Attias and William W. Cohen. Bionlp: Proceedings of the 2012 workshop on biomedical natural language processing. pages 47–55. Association for Computational Linguistics, 2012. 6.3.4

[95] Ndapandula Nakashole, Tomasz Tylenda, and Gerhard Weikum. Fine-grained semantic typing of emerging entities. pages 1488–1497, 2013. 5.1, 6.2.1, 6.3.1, 6.4.1

[96] Roberto Navigli. Word sense disambiguation: A survey. *ACM Computing Surveys*, 41(2), 2009. 6.4.2

[97] Kamal Nigam and Rayid Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, CIKM '00, pages 86–93, New York, NY, USA, 2000. ACM. ISBN 1-58113-320-0. doi: 10.1145/354756.354805. 5.1, 5.5

[98] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39 (2-3):103–134, 2000. 4.1.3, 4.6, 5.1, 6.1.1

[99] Aditya Pal, Nilesh Dalvi, and Kedar Bellare. Discovering hierarchical structure for sources and entities. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013. 1.3, 7.6

[100] Patrick Pantel and Deepak Ravichandran. Automatically labeling semantic classes. In *HLT-NAACL*, 2004. 3, 3.2

[101] Aditya Parameswaran, Hector Garcia-Molina, and Anand Rajaraman. Towards the web of concepts: Extracting concepts from large datasets. volume 3, pages 566–577. VLDB Endowment, 2010. 3.2

[102] D. Pelleg and A. Moore. X-means: Extending K-means with efficient estimation of the number of clusters. In *ICML*, 2000. 4.6

[103] Jing Peng, Costin Barbu, Guna Seetharaman, Wei Fan, Xian Wu, and Kannappan Palaniappan. Shareboost: Boosting for multi-view learning with performance guarantees. In Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 6912 of *Lecture Notes in Computer Science*, pages 597–612. Springer Berlin Heidelberg, 2011. ISBN

978-3-642-23782-9. doi: 10.1007/978-3-642-23783-6_38. 5.5

[104] Taher Mohammad Pilehvar and Roberto Navigli. A robust approach to aligning heterogeneous lexical resources. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 468–478. Association for Computational Linguistics, 2014. 6.4.2

[105] Simone Paolo Ponzetto and Roberto Navigli. Knowledge-rich word sense disambiguation rivaling supervised systems. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1522–1531, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. 6.4.2

[106] E.H. Ramirez, R. Brena, D. Magatti, and F. Stella. Probabilistic metrics for soft-clustering and topic model validation. In *Web Intelligence and Intelligent Agent Technology (WI-IAT)*, 2010. 3.1.2, A.2.2

[107] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 1971. 3.1.2

[108] Joseph Reisinger and Marius Paşca. Latent variable models of concept-attribute attachment. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 620–628. Association for Computational Linguistics, 2009. 7.6

[109] Jason Rennie. 20-newsgroup dataset. *http://qwone.com/ jason/20Newsgroups/*, 2008. 4.2

[110] Alan Ritter, Stephen Soderland, and Oren Etzioni. What is this, anyway: Automatic hypernym discovery. In *AAAI Spring Symposium: Learning by Reading and Learning to Read*, pages 88–93, 2009. 3.2

[111] Sunita Sarawagi. Information extraction. *Foundations and trends in databases*, 1(3): 261–377, 2008. 2.1

[112] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *Advances in Neural Information Processing Systems*, volume 12, pages 582–588. 1999. 4.6

[113] Keiji Shinzato and Kentaro Torisawa. Acquiring hyponymy relations from web documents. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 73–80, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics. 3.2

[114] Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. A co-regularization approach to semi-supervised learning with multiple views. In *ICML Workshop on Learning with Multiple Views*, 2005. 5.5

[115] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems*, 2004. 3.2

[116] Rion Snow, Daniel Jurafsky, and Andrew Y Ng. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 801–808. Association for Computational Linguistics, 2006. 1.3, 7.6

[117] C Studholme. Measures of 3d medical image alignment. *PhD Thesis, University of London, London, UK*, 1997. 3.1.2

[118] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-654-7. doi: 10.1145/1242572.1242667. 6.3.1, 6.3.1, 6.4.2

[119] Partha P Talukdar. Graph-based weakly-supervised methods for information extraction & integration. 2010. 6.4.2

[120] Partha Pratim Talukdar and Koby Crammer. New regularized algorithms for transductive learning. In *Machine Learning and Knowledge Discovery in Databases*, pages 442–457. Springer, 2009. 4.6, 6.3.3

[121] Partha Pratim Talukdar, Joseph Reisinger, Marius Paşca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. Weakly-supervised acquisition of labeled class instances using graph random walks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 582–590, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. 3.2

[122] **Bhavana Dalvi** and William W. Cohen. Very fast similarity queries on semi-structured data from the web. In *SIAM International Conference on Data Mining*, 2013. 4.6, 5.1, 5.5

[123] **Bhavana Dalvi** and William W. Cohen. Multi-view hierarchical semi-supervised learning by optimal assignment of sets of labels to instances. 2015. 1.3, 5, 6.2, 7.3.2

[124] **Bhavana Dalvi** and William W. Cohen. Hierarchical semi-supervised classification with incomplete class hierarchies. In *under preparation*, 2015. 1.3, 7, 8.1

[125] **Bhavana Dalvi**, William W. Cohen, and Jamie Callan. Collectively representing semi-structured data from the web. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-Scale Knowledge Extraction*, pages 7–12. Association for Computational Linguistics, 2012. 4.6, 5.5

[126] **Bhavana Dalvi**, William W. Cohen, and Jamie Callan. Websets: Extracting sets of entities from the web using unsupervised information extraction. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 243–252, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-0747-5. doi: 10.1145/2124295.2124327. 1, 1.4, 3, 3.1.1, 3.1.2, 4.2, 4.6, 5.3.1, 7.6

[127] **Bhavana Dalvi**, William W. Cohen, and Jamie Callan. Classifying entities into an incomplete ontology. In *Proceedings of the 2013 workshop on Automated Knowledge Base Construction*, pages 31–36. ACM, 2013. 1.3, 5.1, 7, 7.3.1, 8.1

[128] **Bhavana Dalvi**, William W. Cohen, and Jamie Callan. Exploratory learning. In *Machine Learning and Knowledge Discovery in Databases*, pages 128–143. Springer, 2013. 1.3, 4, 5.1, 6.5, 7, 7.2.2, 7.2.3, 7.4.2, 8.1, 8.2

[129] **Bhavana Dalvi**, Chenyan Xiong, and Jamie Callan. A language modeling approach to entity recognition and disambiguation for search queries. In *Proceedings of the first International workshop on Entity Recognition & Disambiguation*, pages 45–54. ACM, 2014.

1.3, 6.3.1

[130] **Bhavana Dalvi**, Einat Minkov, Partha P. Talukdar, and William W. Cohen. Automatic gloss finding for a knowledge base using ontological constraints. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 369–378, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3317-7. doi: 10.1145/2684822.2685288. 6.3, 7.3.2, 8.1

[131] Nicolas Usunier, Massih-Reza Amini, and Cyril Goutte. Multiview semi-supervised learning for ranking multilingual documents. *Lecture Notes in Computer Science*, 6913:443–458, 2011. 5.1, 5.5

[132] Benjamin Van Durme and Marius Pasca. Finding cars, goddesses and enzymes: Parametrizable acquisition of labeled instances for open-domain information extraction. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, pages 1243–1248. AAAI Press, 2008. ISBN 978-1-57735-368-3. 3, 3.1.2, 3.2, A.2.4

[133] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained K-means clustering with background knowledge. In *International Conference on Machine Learning*, volume 1, pages 577–584, 2001. 4.6

[134] Chi Wang, Kaushik Chakrabarti, Tao Cheng, and Surajit Chaudhuri. Targeted disambiguation of ad-hoc, homogeneous sets of named entities. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 719–728, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1229-5. doi: 10.1145/2187836.2187934. 6.4.2

[135] Meng Wang, Xian-Sheng Hua, Richang Hong, Jinhui Tang, G-J Qi, and Yan Song. Unified video annotation via multigraph learning. *Circuits and Systems for Video Technology, IEEE Transactions on*, 19(5):733–746, 2009. 6.4.1

[136] Richard C. Wang and William W. Cohen. Automatic set instance extraction using the web. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 441–449, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-45-9. 3.1.2, 5, 3.2, A.1.3

[137] Richard C Wang and William W Cohen. Character-level analysis of semi-structured documents for set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3*, pages 1503–1512. Association for Computational Linguistics, 2009. 3.1.2, 6.3.3

[138] Max Welling and Kenichi Kurihara. Bayesian K-Means as a Maximization-Expectation algorithm. In *SIAM International Conference on Data Mining*, pages 474–478, 2006. 4.6

[139] Robert Wetzker, Carsten Zimmermann, and Christian Bauckhage. Analyzing social bookmarking systems: A del.icio.us cookbook. *Mining Social Data (MSoDa) Workshop Proceedings, ECAI*, 2008. `http://www.dai-labor.de/en/competence_centers/irml/datasets/`. 3.1.2, 1, 2

[140] Derry Wijaya, Partha Pratim Talukdar, and Tom Mitchell. Pidgin: Ontology alignment

using web text as interlingua. In *Proceedings of the 22Nd ACM International Conference on Conference on Information & Knowledge Management*, CIKM '13, pages 589–598, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2263-8. doi: 10.1145/2505515.2505559. 6.3.3

[141] Peter Willett. Recent trends in hierarchic document clustering: a critical review. *Information Processing & Management*, 24(5):577–597, 1988. 7.6

[142] Lin Xiao, Dengyong Zhou, and Mingrui Wu. Hierarchical classification via orthogonal transfer. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 801–808, 2011. 7.6

[143] Chenyan Xiong and Jamie Callan. Esdrank: Connecting query and documents through external semi-structured data. In *International Conference on Information and Knowledge Management*, 2015. 6.3.1, 6.4.2

[144] Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning. *CoRR*, 2013. 5.1, 5.5

[145] Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. Textrunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 25–26. Association for Computational Linguistics, 2007. 3.2, 4.6

[146] Mo Yu, Shu Wang, Conghui Zhu, and Tiejun Zhao. Semi-supervised learning for word sense disambiguation using parallel corpora. In *Eighth International Conference on Fuzzy Systems and Knowledge Discovery*, 2011. 6.4.2