

**UNIVERSAL SCHEMA FOR
KNOWLEDGE REPRESENTATION
FROM TEXT AND STRUCTURED DATA**

A Dissertation Presented

by

LIMIN YAO

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 7, 2014

Computer Science

© Copyright by Limin Yao 2012

All Rights Reserved

**UNIVERSAL SCHEMA FOR
KNOWLEDGE REPRESENTATION
FROM TEXT AND STRUCTURED DATA**

A Dissertation Presented

by

LIMIN YAO

Approved as to style and content by:

Andrew McCallum, Chair

Benjamin Marlin, Member

Daniel Sheldon, Member

Rajesh Bhatt, Member

Luke Zettlemoyer, Member

Lori Clarke, Chair
Computer Science

ABSTRACT

UNIVERSAL SCHEMA FOR KNOWLEDGE REPRESENTATION FROM TEXT AND STRUCTURED DATA

December 7, 2014

LIMIN YAO

B.Sc., XI'AN JIAOTONG UNIVERSITY

M.Sc., TSINGHUA UNIVERSITY

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Andrew McCallum

In data integration we transform information from a source into a target schema. A general problem in this task is loss of fidelity and coverage: the source expresses more knowledge than that can be fit into the target schema, or knowledge that is hard to fit into any schema at all. This problem is taken to an extreme in information extraction (IE) where the source is natural language—one of the most expressive forms of knowledge representation. To address this issue, one can either automatically learn a latent schema emergent in text (a brittle and ill-defined task), or manually define schemas. We propose instead to store data in a probabilistic representation of *universal schema*. This schema is simply the union of all source schemas, and we learn how to predict the cells of each source relation in this union. For example, we could store Freebase relations and relations that are expressed by natural language

surface patterns. To populate such a database of universal schema, we present matrix factorization models that learn latent embedding vectors for entity tuples and relations.

We show that such latent models achieve substantially higher accuracy than a traditional classification approach on New York Times and Freebase data. Besides binary relations, we also use universal schema for unary relations, i.e., entity types. We also explore various facets of universal schema matrix factorization models on a large-scale web corpus, including implicature among the relations. We also evaluate our approach on the task of question answering using features obtained from universal schema, achieving state-of-the-art accuracy on a benchmark dataset.

TABLE OF CONTENTS

	Page
ABSTRACT	iv
LIST OF TABLES	x
LIST OF FIGURES	xiv
 CHAPTER	
1. INTRODUCTION	1
1.1 Contributions	2
1.2 Declaration of Previous Work	3
2. BACKGROUND	4
2.1 Relations	4
2.2 Mentions	4
2.3 Dependency Path	5
2.4 Relation Extraction	5
2.5 Entity Types	6
2.6 Selectional Preferences	6
2.7 Data Preprocessing	6
2.8 Entity Linking	7
2.9 Graphical Models	8
2.9.1 Discriminative Models	8
2.9.2 Generative Models	9
3. DISTANT SUPERVISION FOR RELATION EXTRACTION	12
3.1 Distant Supervision	12
3.2 Modeling Relations and Their Mentions	13
3.2.1 Models	14
3.2.2 Experiments	15

3.3	Joint Inference for Entity and Relation Extraction	17
3.3.1	Model	19
3.3.2	Learning and Inference	21
3.3.3	Experiments	22
3.3.3.1	Wikipedia data	24
3.3.3.2	New York Times data	26
3.4	Related Work	29
3.4.1	Supervised Relation Extraction	29
3.4.2	Distant Supervision	29
3.4.3	Joint Entity and Relation Extraction	30
3.5	Conclusion	30
4.	UNSUPERVISED RELATION EXTRACTION USING GENERATIVE MODELS	32
4.1	Introduction	32
4.2	Models	34
4.2.1	Rel-LDA Model	34
4.2.2	Rel-LDA1 model	36
4.2.3	Type-LDA model	37
4.3	Experiments	39
4.3.1	Relations discovered by different models	39
4.3.2	Distant Supervision based Relation Extraction	42
4.3.3	Comparing against USP	46
4.4	Related Work	47
4.5	Conclusion	48
5.	UNSUPERVISED RELATION DISCOVERY WITH SENSE DISAMBIGUATION	49
5.1	Introduction	49
5.2	Our Approach	51
5.2.1	Sense Disambiguation	52
5.2.2	Hierarchical Agglomerative Clustering	54
5.3	Experiments	55

5.3.1	Feature Extraction	55
5.3.2	Sense clusters and relation clusters	57
5.3.3	Baselines	58
5.3.4	Automatic Evaluation against Freebase	59
5.3.5	Path Intrusion	60
5.3.6	Error Analysis	62
5.4	Related work	63
5.5	Conclusion	64
6.	UNIVERSAL SCHEMA FOR ENTITY TYPE CLASSIFICATION	65
6.1	Introduction	65
6.2	Factorization Models	70
6.2.1	Neighbor Model	73
6.3	Experiments	74
6.3.1	Data Sets	74
6.3.2	Baselines	77
6.3.3	Pattern Analysis on NYT data	77
6.3.4	Closed Set Evaluation	78
6.3.5	Evaluation on WikiLinks	80
6.3.6	Parameter Selection	82
6.4	Related Work	82
6.5	Conclusion	83
7.	UNIVERSAL SCHEMA FOR RELATION EXTRACTION	84
7.1	Introduction	84
7.2	Models	86
7.2.1	Matrix Factorization	86
7.2.2	Neighbor Model	88
7.2.3	Entity Model	88
7.2.4	Alternative Training Objectives	89
7.3	Evaluation	90
7.3.1	Data	91
7.3.2	Evaluation Measures	92
7.3.3	Baselines	93
7.3.4	Ranking based Evaluation	93

7.3.5	Classification based Evaluation	96
7.3.6	Integrating entity types	96
7.4	Exploration on Facets of Universal Schema	98
7.4.1	Does more training data lead to better performance?	99
7.4.2	Does the number of components matter?	100
7.4.3	Does the non-convex objective affect the performance?	100
7.4.4	How does coreference affect the final performance?	100
7.4.5	Can our approach discover implications among relations?	101
7.4.6	Error Analysis	105
7.5	Related Work	108
7.6	Conclusion	110
8.	QUESTION ANSWERING FROM FREEBASE	112
8.1	Question Answering System	112
8.2	Experiments	114
8.2.1	Error Analysis	115
8.3	Related Work	116
8.4	Conclusion	117
9.	CONCLUSIONS AND FUTURE DIRECTIONS	118
	BIBLIOGRAPHY	121

LIST OF TABLES

Table	Page
3.1 The statistics of held-out evaluation on Wikipedia and New York Times.	25
3.2 Average and weighted (w) average precision over frequent relations for New York Times and Wikipedia data, based on manual evaluation.	26
3.3 Precision at 50 for the most frequent relations on New York Times	28
4.1 The features of tuple ‘(Gamma Knife, made by, Elekta)’ in sentence “Gamma Knife, made by the Swedish medical technology firm Elekta, focuses low-dosage gamma radiation ...”	35
4.2 The notation used in our models	35
4.3 Clustering quality evaluation (%). Recall is measured against Freebase. Precision is measured according to human annotators	40
4.4 The path, source and destination arguments of some relations found by Rel-LDA1.	42
4.5 The entity clusters found by Type-LDA	42
4.6 Precision (%) of some frequent relations	44
4.7 Manual evaluation, Precision and recall of some frequent relations	45

5.1	Example sense clusters for pattern “A play B” produced by sense disambiguation. For each sense, we randomly sample 5 entity pairs. We also show top features for each sense. Each row shows one feature type, where “num” stands for digital numbers, and prefix “l:” for source argument, prefix “r:” for destination argument. Some features overlap with each other. We manually label each sense for easy understanding. We can see the last two senses are close to each other. For two theme features, we replace the theme number with the top words. For example, the document theme of the first sense is Topic30, and Topic30 has top words “sports”. The lower four rows are four types of features: document theme, sentence theme, lexical words and entity names.	55
5.2	Example semantic relation clusters produced by our approach. For each cluster, we list the top paths in it, and each is followed by “:number”, indicating its sense obtained from sense disambiguation. They are ranked by the number of entity pairs they take. The column on the left shows sense of each relation. They are added manually by looking at the sense numbers associated with each path.	56
5.3	Pairwise and B^3 evaluation for various systems. Since our systems predict more fine-grained clusters than Freebase, the recall measure is underestimated.	61
5.4	A path intrusion task. We show 5 paths and ask the annotator to identify one path which does not belong to the cluster. And we show one example sentence for each path. The entities (As and Bs) in the sentences are bold. And the italic row here indicates the intruder.	61
5.5	Results of intruding tasks of all systems.	62
6.1	Some entities, observed types and predicted ones by our system. We can describe an entity in any granularity based on the patterns or types from ontologies. The patterns are translated from dependency parsing paths as described in §6.3.	70
6.2	Top similar patterns to the target queries.	78
6.3	Top ranked patterns learned by the baseline classifiers. Not all patterns can imply the target patterns. The patterns are not as diverse as patterns learned by our approach.	79

6.4	Performance on predicting Freebase entity types on a closed data set.	80
6.5	Performance on predicting Freebase entity types on WikiLink.	80
6.6	F1 measure on some human annotated fine-grained types. We take these types as representatives of pattern based types.	81
7.1	Average and (weighted) Mean Average Precisions for Freebase relations based on pooled results. The # column shows the number of true facts in the pool. NFE is statistically different to all but NF and F according to the sign test. Bold faced are winners per relation, italics indicate ties.	95
7.2	Performance on predicting Freebase relations in universal schema. +Unary indicates adding predicted pattern based types as features. +Unary(FB) for adding predicted Freebase types	97
7.3	Selectional preferences learned by our model	98
7.4	Statistics of processed data.	99
7.5	Performance variations of relation extraction as the amount of training data increases.	99
7.6	Performance variations of relation extraction as the number of dimensions varies.	100
7.7	Performance variations of relation extraction for different initializations.	100
7.8	Performance variations of relation extraction as we use different coreference approaches.	101
7.9	Examples of asymmetric implication pairs.	102
7.10	Top narrower relations for some broader relations.	103
7.11	The number of relations predicted as true based on only one observed relation. Broad relations (i.e., “player”) turn on more predictions; narrow relations (e.g., “pitcher”) turn on fewer predictions.	104

8.1	Combined features from question and candidates. ‘ctype’ stands for entity types of a candidate. ‘qtype’ stands for entity types of a question topic. ‘qword’ and ‘qfocus’ are similar to those defined in [Yao and Durme, 2014]	114
8.2	Performance on WebQuestions: a question answering data set annotated with answers from Freebase.	115

LIST OF FIGURES

Figure	Page
2.1 Latent Dirichlet Allocation model. z represents a topic, w represents a word token in a document, Z represents the number of topics, and D represents the number of documents.	10
3.1 Overview of distant supervision.	13
3.2 Factor Graph for joint relation mention prediction and relation type identification. For each pair of entities that are mentioned together in at least one sentence we create one relation variable (the top variable here). For each of the pairs of entity mentions that appear in a sentence we create one relation mention variable, and connect it to the corresponding relation variable. Note that relation variables for different pairs of entities are considered to be independent.	14
3.3 The recall and precision curves for the held out evaluation of three approaches: distant supervision, joint supervision, and at-least-once supervision	16
3.4 Precision at \mathbf{K} for manually evaluated predictions	17
3.5 Factor Graph of our model that captures selectional preferences and functionality constraints. For readability we only label a subsets of equivalent variables and factors. Note that the graph shows an example assignment to variables.	19
3.6 Precision-recall curves for various setups in Wikipedia held-out setting.	25
3.7 Precision-recall curves for isolated, pipeline and joint approaches in New York Times held-out setting.	27
4.1 Rel-LDA model. Shaded circles are observations, and unshaded ones are hidden variables. A document consists of N tuples. Each tuple has a set of features. Each feature of a tuple is generated independently from a hidden relation variable r	36

4.2	Type-LDA model. Each document consists of N tuples. Each tuple has a set of features, relation level features f and entity level features of source argument f_s and destination argument f_d . Relation level features and two hidden entity types T_1 and T_2 are generated from hidden relation variable r independently. Source entity features are generated from T_1 and destination features are generated from T_2	38
5.1	Sense-LDA model.	54
6.1	Overview of our system.	68
7.1	Averaged 11-point precision recall curve for surface pattern relations.	94
7.2	Different factorization models for relation extraction. They are different in terms of how they model relations and entities. The first model represents relations as matrices. The remaining models represent relations as vectors. All the first three models represent each entity as a vector. The fourth model, our factorization model, represents each entity pair as a vector. Except for the first model, all the models are scalable.	110

CHAPTER 1

INTRODUCTION

Natural language is a highly expressive representation of knowledge. Yet, for many tasks knowledge bases are more suitable, as they support more effective querying, question answering and data mining. But knowledge bases usually have a pre-defined schema, and they can only capture so much of the information natural language can express. For example, Freebase [Bollacker et al., 2008] captures the content of Wikipedia to some extent, but has no criticized(Person,Person) relation and hence cannot answer a question like “Who criticized George Bush,” even though partial answers are expressed in Wikipedia. This makes the database schema a major bottleneck in information extraction (IE). From a more general point of view, data integration always suffers from schema mismatch between knowledge source and knowledge target, for example, integrating natural language text with Freebase.

To overcome this problem, one could attempt to manually extend the schema whenever needed, but this is a time-consuming and expensive process. Alternatively, in the case of IE, we can automatically induce latent schemas from text, but this is a brittle, ill-defined and error-prone task. This thesis presents a third alternative: sidestep the issue of incomplete schemas altogether, by simply combining the relations of all knowledge sources into what we refer to as a universal schema. So universal schema is the union of all relations seen among natural language surface patterns and other structured knowledge sources. We generalize these source relations by learning implications among them using matrix factorization. Experimental results on New York Times data show that, comparing against recent distant supervision baseline

systems, relation extraction using universal schema achieves better performance. We explore various facets of universal schema matrix factorization models on a large-scale web corpus, including implicature among the relations. We also evaluate our approach on the task of question answering using features obtained from universal schema, achieving state-of-the-art accuracy on a benchmark dataset.

In this thesis, we first introduce background knowledge in Chapter 2. After that, we introduce alternative approaches to relation extraction: distant supervision in Chapter 3, unsupervised relation discovery with topic models in Chapter 4, and pattern sense disambiguation in Chapter 5. Then we describe our core approach universal schema, starting with universal schema for entity types in Chapter 6, and exploring universal schema for relation extraction in Chapter 7. We also describe a question answering application using universal schema in Chapter 8. Finally we conclude and list future research directions in Chapter 9.

1.1 Contributions

This thesis mainly describes our work on building knowledge bases of entities and relations using different approaches. Here are my contributions.

- Present universal schema for representing entities (6) and relations (7), allowing surface patterns to stand for themselves without information loss, and employ matrix factorization for learning implications among relations (§7.2).
- Explore different applications of universal schema, including relation instance prediction, question answering. (§7.3, 8).
- Develop relation extraction models based on distant supervision, addressing the challenge that not all relation mentions in the text corpus express the relation from the distant supervision source, and incorporating selectional preferences to improve the performance (3).

- Develop generative models for unsupervised relation discovery, assuming each relation tuple is generated from a relation topic, and dealing with the ambiguity of pattern senses by clustering entity pairs of each pattern (4, 5).

1.2 Declaration of Previous Work

I declare my previous work and collaborations with other researchers. All of them are directed by my advisor Andrew McCallum.

- Work on modeling the mismatching between the knowledge base and the text corpus in distant supervision (§3.2) is in collaboration with Sebastian Riedel, published as [Riedel et al., 2010].
- Work on joint modeling of entity and relation types in distant supervision (§3.3) is in collaboration with Sebastian Riedel, published as [Yao et al., 2010].
- Work of developing topic models for unsupervised relation extraction (4) is published as [Yao et al., 2011], and in collaboration with Aria Haghighi and Sebastian Riedel.
- Work of disambiguating pattern senses using topic model (5) is published as [Yao et al., 2012b], and in collaboration with Sebastian Riedel.
- Work of developing efficient inference algorithms for topic models is in collaboration with David Mimno, published as [Yao et al., 2009].
- Work of universal schema for entity types (6) is published as [Yao et al., 2013], and in collaboration with Sebastian Riedel.
- Work of universal schema (7) for relation extraction is in collaboration with Sebastian Riedel, published as [Yao et al., 2012a, Riedel et al., 2013].

CHAPTER 2

BACKGROUND

This chapter briefly introduces the terminology for defining the tasks, and machine learning models for addressing the tasks.

2.1 Relations

Relation extraction deals with extraction of relationships among entities. Example entities include the company founder **BILL GATES**, the company **MICROSOFT**, and the country **USA**. A *relation* R is a set of tuples over entities. We call $R(e_1, \dots, e_n)$ with tuple $(e_1, \dots, e_n) \in R$ a *relation instance*. It denotes the membership of the tuple in the relation R . For example, **founded** (**BILL GATES**, **MICROSOFT**) is a relation instance denoting that **BILL GATES** and **MICROSOFT** are related in relation **founded**.

2.2 Mentions

In natural language, text spans of tokens are used to refer to entities. We call such spans *entity mentions*. Consider, for example, the sentence snippet: “Political opponents of President **Evo Morales** of **Bolivia** have in recent days stepped up” Here “Evo Morales” is an entity mention of president **EVO MORALES**, and “Bolivia” a mention of the country **BOLIVIA** he is the president of.

People often express relations between entities in natural language texts by mentioning the participating entities in specific syntactic and lexical patterns. Any tuple of mentions of entities (e_1, \dots, e_n) in a sentence is defined as a *candidate mention tuple*.

Note that tuples across sentences are not considered in our work. If such a candidate expresses the relation R , then it is a *relation mention* of the relation instance $R(e_1, \dots, e_n)$. In this thesis, we mostly work on binary relation instances. Usually our basic data unit is a triple $(e_1, \text{pattern}, e_2)$ that has two arguments e_1 and e_2 , and a pattern that expresses the relationship between them.

2.3 Dependency Path

As described in §2.2, relations between entities are expressed by specific syntactic and lexical patterns. To obtain such patterns, we first parse sentences to get dependency trees. In a dependency tree, each word is a node and a dependency relation between two words is an edge. For binary relations, we employ dependency paths that connect entity pairs to represent relations. A dependency path is a concatenation of dependency relations (edges) and words (nodes) along a path in a dependency tree [Lin and Pantel, 2001]. For instance, the sentence “John Lennon was born in Liverpool” would yield the relation tuple (JOHN LENNON, [\leftarrow nsubjpass \leftarrow bear \rightarrow prep \rightarrow in \rightarrow pobj \rightarrow], LIVERPOOL). This relation instance reflects a semantic *bornIn* relation between the JOHN LENNON and LIVERPOOL entities. The dependency path in this example corresponds to the “X was born in Y” textual expression. Through this proposal, we mainly use dependency paths to represent relationships of entity pairs.

2.4 Relation Extraction

The goal of relation extraction is to discover the truth about underlying relations among entities—to discover the semantic meanings of relations irrespective of how they are expressed. For example, in supervised and weakly supervised approaches, relation types are defined to represent relations. In unsupervised approaches, a set of surface patterns falling in one cluster represents a relation. In universal schema, each

surface pattern or a relation type represents a relation. The underlying semantics of these relations are expressed by implications among them.

2.5 Entity Types

An entity can be categorized into one or several *entity types*. For example, BILL GATES is a **person**, and a **founder** of a company. Entity types correspond to the special case of relations with arity one, and we usually name them unary relations.

Discovering entity types can help us understand the concepts of specific domains. For relation extraction, we care about entity types because they are useful for extracting binary relations due to selectional preferences—see §2.6.

We care about entity types due to their importance in downstream applications: if consumers of our extracted facts know the type of entities, they can find them more easily, visualize them more adequately, and perform operations specific to these types (write emails to persons, book a hotel in a city, etc.).

2.6 Selectional Preferences

There are type constraints between relations and entities, known as *selectional preferences*. Specifically, relations require, or prefer, their arguments to be of certain types. For example, the **nationality** relation requires the first argument to be a **person**, and the second to be a **country**. In relation extraction, we expect that modeling these type constraints can improve the performance [Roth and Yih, 2007, Pantel et al., 2007].

2.7 Data Preprocessing

We use the following procedure to generate candidate relation tuples: first we use the Stanford named entity recognizer (NER) [Finkel et al., 2005] to find entity mentions in the corpus. The NER tagger segments each document into sentences and

classifies each token into four categories: PERSON, ORGANIZATION, LOCATION and NONE. We treat consecutive tokens that share the same category as single entity mention.

We extract a pair of entity mentions that occur in the same sentence as one candidate tuple, and extract a set of features from the sentence for relation extraction. The features are similar to those used in [Mintz et al., 2009]: lexical, part-of-speech (POS), named entity and dependency path features, as explained in §2.3. We use the openNLP POS tagger¹ to obtain POS tags and employ the MaltParser [Nivre et al., 2004] for dependency parsing.

Our entity type extraction uses the following features: the surface form of the entity, the POS sequence, the head of the entity in the dependency parse tree, the named entity tag, and the left and right words to the current entity mention phrase.

2.8 Entity Linking

Most of our approaches discussed in this thesis require linking entities from a text corpus (e.g., NYT articles) to a knowledge base (e.g., Freebase). For example, most approaches learn models that leverage co-occurrences among multiple relation mentions of the same entity pair. To find these multiple mentions, entity resolution is a prerequisite. However, entity resolution is not our main focus in this thesis, and there are many complex alternative methods from which to choose. To ease reproducibility and demonstrate the robustness of our approaches we use simple string matching to link entity mentions in NYT documents to Freebase entities. Specifically, if the Freebase entity has the same string form as the entity mention, we link them as one entity. If multiple Freebase entities have the same string form, we use the most popular one. For example, “Canada” could be a *country*, or a kind of *wine*. We use

¹available at <http://opennlp.sourceforge.net/>

country. This method has been employed by many researchers [Mintz et al., 2009, Lin and Pantel, 2001, Yao et al., 2010]. Integrating relation extraction with more complex entity resolution systems is a topic for future work.

2.9 Graphical Models

Graphical models have been proven useful for formalizing learning tasks. This section describes graphical model representation and how to learn parameters for these models. In graphical models, we usually formalize a task as predicting the values of variables. There are observed variables and hidden variables. Observed variables capture the information of observed data. Hidden variables capture the latent structure of the data. Usually we predict hidden variables based on observed variables. In this thesis, we use \mathbf{Y} to represent hidden variables and \mathbf{X} to represent observed variables. We note the values of hidden variables as labels, and instances of observed variables as data, observations and evidence. One can categorize graphical models into two categories in terms of how they define their objectives as functions of observed and hidden variables. Discriminative models define the objectives as the conditional probabilities of hidden variables given observed variables, while generative models define the objectives as the joint probabilities of observed and hidden variables.

2.9.1 Discriminative Models

In this thesis we mainly use exponential family models. The models define the conditional probabilities as

$$p(\mathbf{Y}|\mathbf{X}) = \frac{\exp(\vec{\lambda}\mathbf{F}(\mathbf{X}, \mathbf{Y}))}{\sum_{\mathbf{Y}'} \exp(\vec{\lambda}\mathbf{F}(\mathbf{X}, \mathbf{Y}'))}$$

In these models, $\mathbf{F}(\mathbf{X}, \mathbf{Y})$ represents features defined over hidden and observed variables. For example, if a word “baseball” occurs in a document where “sports” is a possible label, we can define a binary feature as (observation=baseball, label=sports).

We employ maximum log likelihood estimation to learn the parameters $\vec{\lambda}$. Given a list of pairs (\vec{x}_i, \vec{y}_i) , we define the objective function as:

$$\sum_i \log p_i(\vec{y}_i|\vec{x}_i) - \frac{1}{2\sigma^2} \|\vec{\lambda}\|^2$$

In this formula, the first term is the log likelihood of the data and the second one is the regularization term. In this thesis we mainly use L2 regularization. Other regularizations are possible too. To maximize the objective function, we use the gradient descent optimization method. We derive the gradients with respect to the parameters:

$$\nabla \vec{\lambda} = \mathbf{F}(\mathbf{X}, \mathbf{Y}) - \mathbf{E}_{p_{\vec{\lambda}}}(\mathbf{X}, \mathbf{Y})$$

The gradients have two terms, the constraints term and the expectation term. The constraints are the statistics obtained from the training data. The expectation term calculates the expectations of the features under the assumption that the data follows the distribution defined by the model. The objective is maximized when the constraints match the expectations. Intuitively, one set of parameters define a distribution, and the goal of learning is to find one distribution in the whole parameter space that can explain the data well.

2.9.2 Generative Models

Discriminative models are usually used in supervised settings, i.e. when we have annotated relations available. When we do not have labeled data, we can use unsupervised models. Generative models, mainly topic models are popular unsupervised models. Generative models assume the data is generated from a probabilistic process. Take one document as an example: in a simple case, we can assume that each word is generated independently from a probabilistic distribution.

One example of a generative process is the standard topic model named Latent Dirichlet Allocation (LDA) [Blei et al., 2003]. This model assumes that a document

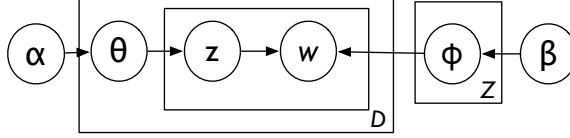


Figure 2.1. Latent Dirichlet Allocation model. z represents a topic, w represents a word token in a document, Z represents the number of topics, and D represents the number of documents.

has a distribution over topics, each topic has a distribution over words in a vocabulary and each word is drawn from a topic. Figure 2.1 shows the graphical representation of this model.

Here is the generative process in detail:

1. For each topic z , generate a distribution ϕ_z over each word from $\text{Dir}(\beta)$
2. For each document d , generate its topic distribution θ_d from $\text{Dir}(\alpha)$
3. For each word in a document d ,
 - Draw a topic z from $\text{Multi}(\theta_d)$,
 - Generate a word w from $\text{Multi}(\phi_z)$ independently

There are various ways to estimate the document topic distribution θ_d and the topic word distribution ϕ_z . We use Gibbs sampling in this thesis. We iteratively sample a topic for each word in each document based on current parameters Θ and Φ , and update parameters based on the current topic assignments.

$$\begin{aligned}
 p(z|w) &\propto p(z|d)p(w|z) \\
 &\propto (\alpha_z + n_{z|d}) \frac{\beta_w + n_{w|z}}{\sum_w (\beta_w + n_{w|z})}
 \end{aligned}$$

$p(z|d)$ and $p(w|z)$ are estimated as follows:

$$\theta_d = \frac{\alpha + n_{z|d}}{\sum_z (\alpha + n_{z|d})}$$
$$\phi_{zw} = \frac{\beta_w + n_{w|z}}{\sum_w (\beta_w + n_{w|z})}$$

Topic models are successful in modeling documents. We will adapt topic models for unsupervised relation extraction.

CHAPTER 3

DISTANT SUPERVISION FOR RELATION EXTRACTION

In distant supervision, we align knowledge bases with text data to set up the training source. In this chapter, we first review distant supervision. After that, we point out challenges in distant supervision and present our approach for explicitly modeling mentions. Last, we extend the distant supervision approach to model entities and relations jointly.

3.1 Distant Supervision

In relation extraction we often encounter a lack of explicitly annotated text, but an abundance of structured data sources such as company databases or collaborative knowledge bases like Freebase are available. In distant supervision, we align relation instances from structured data sources with a text corpus in which they are mentioned to set up the training data [Mintz et al., 2009, Bunescu and Mooney, 2007, Bellare et al., 2007]. The knowledge base plays the role of a distant supervisor for training a relation extractor.

This approach performs relation extraction within a larger scope that considers mentions across documents and takes advantage of redundancy. Often facts are mentioned in several sentences and documents. It may be difficult to parse some of these mentions, or they use unseen patterns. However, the more mentions we consider, the higher the probability that it is easier to parse a mention, and encounter a pattern that we have seen in the training data.

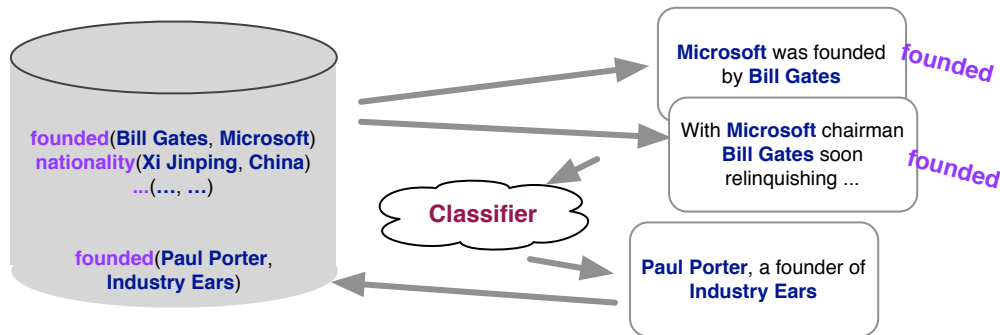


Figure 3.1. Overview of distant supervision.

Figure 3.1 shows an overview process of distant supervision. Starting with a knowledge base of relation instances, we find mentions of them in the text corpus. This requires co-reference and entity linking, we use a simple string matching approach for both of these tasks §2.8. After linking relation instances to sentences in the text, for each entity pair, we collect features from all the sentences that mention them. We train a multi-class classifier for relation extraction. At test time, given new text documents, we first identify candidate tuples, i.e., entity pairs that occur in the same sentence. We then extract a feature vector from all sentences that mention the entity pair. Finally we apply our trained classifier to categorize this pair into one relation type. To expand our knowledge base, we can add this newly predicted relation instance into the knowledge base.

3.2 Modeling Relations and Their Mentions

One challenge in distant supervision arises from the mismatch between relation instances in the knowledge base and their mentions in the form of textual expressions. That is, sometimes two entity mentions appear together in a sentence, but the sentence is not expressing the relation that appears in the KB.

We present a novel approach to distant supervision that can alleviate this problem based on the following two ideas. First, we use a factor graph to explicitly model the

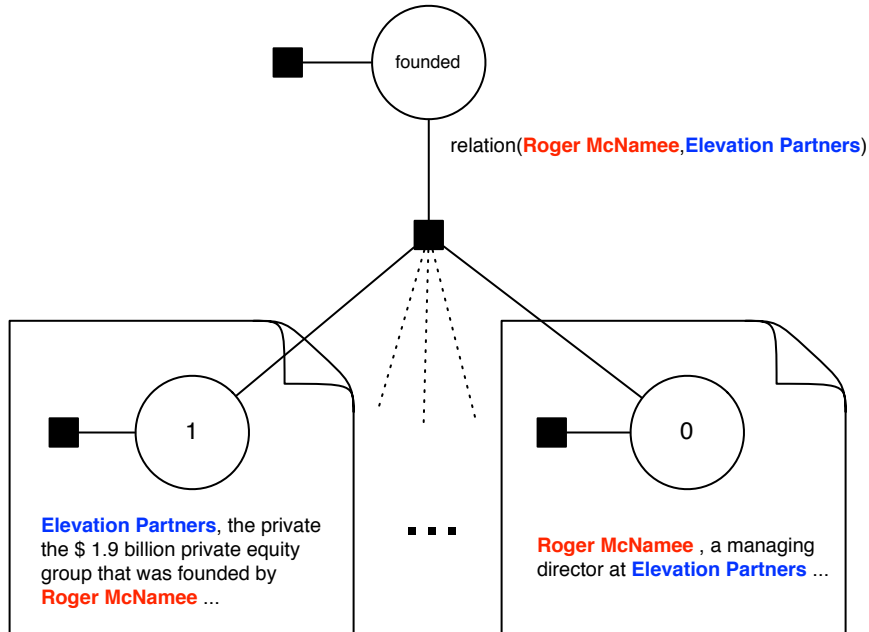


Figure 3.2. Factor Graph for joint relation mention prediction and relation type identification. For each pair of entities that are mentioned together in at least one sentence we create one relation variable (the top variable here). For each of the pairs of entity mentions that appear in a sentence we create one relation mention variable, and connect it to the corresponding relation variable. Note that relation variables for different pairs of entities are considered to be independent.

decision whether two entities are related, and the decision whether this relation is mentioned in a given sentence. Second, we apply constraint-driven semi-supervision to train this model without any knowledge about which sentences express the relations in our training KB [Riedel et al., 2010] .

3.2.1 Models

For modeling, we employ the following expressed-at-least-once assumption: If two entities participate in a relation, at least one sentence that mentions these two entities express that relation.

Figure 3.2 shows the factor graph of our model. Our model connects a relation variable Y for two entities with a set of binary relation mention variables Z_i . Z_i is true

if and only if mention i is indeed expressing the relation Y between the two entities. Crucially, the relation mention variables Z_i s are unobserved at training time: we only know that a relation is expressed at least once, but not in which sentences. During training, the model prefers at least one Z_i is active if Y is true. In Figure 3.2, we show two example mention variables. For each relation mention, we collect features from the sentence, such as the dependency path between the two entities (§2.7).

Our model is implemented in FACTORIE [McCallum et al., 2009], a probabilistic programming language that simplifies the construction process, as well as inference and learning.

3.2.2 Experiments

We carry out experiments on New York Times articles, and use Freebase as distant supervision source.

We follow [Mintz et al., 2009] and perform two types of evaluation: held-out and manual. In both cases we have a training and a test corpus of documents, and training and test sets of entities. For held-out evaluation we split the set of entities in Freebase into training and test sets. For manual evaluation we use all Freebase entities during training. For testing we use all entities that appear in the test document corpus. This setting is used again in distant supervision experiments in the following chapters.

In Figure 3.3 we compare the precision and recall curve for the baseline distant-supervision model (distant), the supervised joint model (joint) and the distant model with expressed-at-least-once assumption (at-least-once). The joint model favors assigning 1 to each Z_i if Y is true, otherwise prefers assigning each Z_i to 0. The curve is constructed by ranking predicted relation instances using their log linear score. For the distant supervision baseline this score is first normalized by the number of mentions. We traverse this list from high score to low score, and measure precision and recall at each position. We can see that the model with expressed-at-least-once

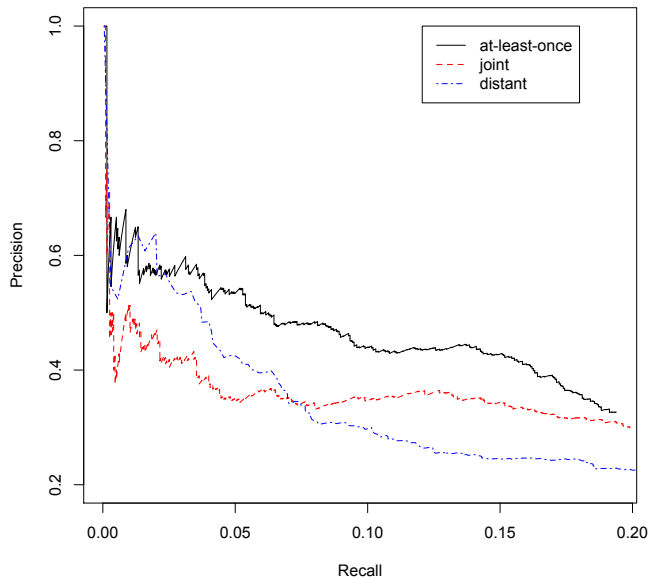


Figure 3.3. The recall and precision curves for the held out evaluation of three approaches: distant supervision, joint supervision, and at-least-once supervision

assumption is consistently outperforming the distant supervision baseline and the supervised joint model. This suggests that the at-least-once model has the best sense of how relations that are already contained in Freebase are expressed in NYT data.

Figure 3.4 shows the manual evaluation results for the three approaches. We first note that the precision is much higher for manual evaluation than for held-out evaluation. This shows that false negatives in Freebase are an issue when doing held-out evaluation. Many of the false positives we predict are in fact true relation instances that just do not appear in Freebase.

For manual evaluation, the at-least-once model is still the clear winner. At $K = 1000$ we observe a precision of 91% for at-least-once supervision and 87% for distant supervision. This amounts to an error reduction rate of 31%. The sign test shows that the at-least-once model is significantly better than the distant supervision model, with $p \ll 0.05$. We also note that despite using the same assumption, the joint

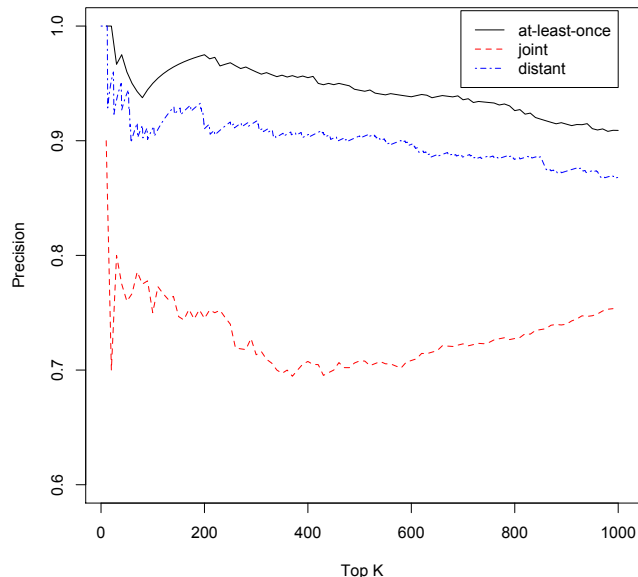


Figure 3.4. Precision at **K** for manually evaluated predictions

model performs much worse than the distant supervision approach in this scenario. Learning a model of relations and mentions is inherently more difficult. Using a wrong assumption will hence more likely hurt performance.

3.3 Joint Inference for Entity and Relation Extraction

In relation extraction, *selectional preferences* describe that certain relations can only hold between particular entity types. For example, for relation `nationality`, the first argument should be a `person` and the second a `country`. These constraints can correct some predictions in relation extraction. In this section, we present how to incorporate them in cross document relation extraction under distant supervision.

A simple way to is to use a pipeline: first predict entity types, and then condition on these when predicting relations. However, this neglects the fact that relations could as well be used to help entity type prediction.

While there is some existing work on enforcing such constraints in a joint fashion [Roth and Yih, 2007, Kate and Mooney, 2010, Riedel et al., 2009], they are not directly applicable in distant supervision. The difference is the amount of facts they take into account at the same time. They focus on single sentence extractions, and only consider very few interacting facts. This allows them to work with exact optimization techniques such as (Integer) Linear Programs and still remain efficient.¹ However, when working on a sentence level they fail to exploit the redundancy present in a corpus. Moreover, the fewer facts they consider at the same time, the lower the chance that some of these will be incompatible, and that modelling compatibility will make a difference.

In this work we present a novel approach that enforces selectional preferences in distant supervision. It is based on an undirected graphical model in which variables correspond to facts, and factors between them measure compatibility. In order to scale up, we run an efficient Gibbs-Sampler at inference time, and train our model using SampleRank [Wick et al., 2009]. In practice this leads to a runtime behaviour that is linear in the size of the corpus. In our experiments, 200K documents take less than three hours for training and testing.

For evaluation we consider two scenarios. First we follow Mintz et al. [2009], use Freebase as a source of distant supervision, and employ Wikipedia as source of unlabelled text—we will call this an in-domain setting. This scenario is somewhat artificial in that Freebase itself is partially derived from Wikipedia, and in practice we cannot expect text and training knowledge base to be so close. Hence we also evaluate our approach on the New York Times corpus (out-of-domain setting).

For in-domain data we make the following finding. When we compare to an isolated baseline that makes no use of entity types, and our joint model improves

¹The pyramid algorithm of Kate and Mooney [2010] may scale well, but it is not clear how to apply their scheme to cross-document extraction.

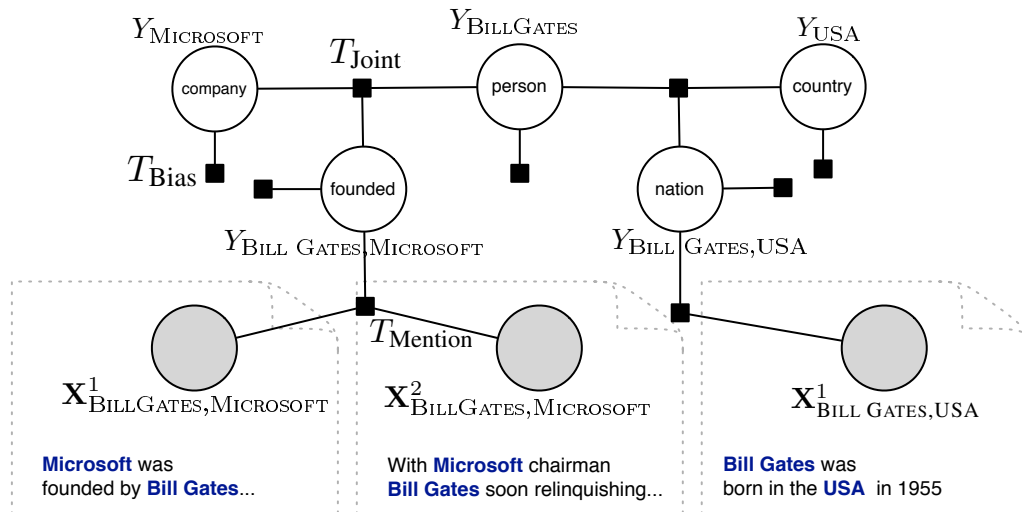


Figure 3.5. Factor Graph of our model that captures selectional preferences and functionality constraints. For readability we only label a subsets of equivalent variables and factors. Note that the graph shows an example assignment to variables.

average precision by 4%. However, it does not outperform a pipelined system. In the out-of-domain setting, our collective model substantially outperforms both other approaches. Compared to the isolated baseline, we achieve a 15% increase in precision. With respect to the pipeline approach, the increase is 13%.

3.3.1 Model

In this section we describe our approach. It is based on Conditional Random Field (CRF), represented as factor graphs, in which variables correspond to entity types and relation types, and factors between them measure compatibility. CRFs are a natural fit for this task: they allow us to capture correlations in an explicit fashion, and to incorporate overlapping input features from multiple documents.

Figure 3.5 shows the factor graph of our model. The hidden output variables of our model are \mathbf{Y} s. We have one relation variable for each candidate tuple, and one entity variable for each entity. Relation variables can take values from the set

of relation types, while entity variables take values from the set of entity types. See example relation variables in figure 3.5.

The observed input variables \mathbf{X} consist of a family of variables for each candidate tuple. Each candidate tuple may have multiple relation mentions. Each variable stores relevant observations we make for the i -th candidate relation mention in the corpus. For example, $\mathbf{X}_{\text{BILL GATES, MICROSOFT}}^1$ in figure 3.5 would contain the pattern “[M2] was founded by [M1].”

Our conditional probability distribution over variables \mathbf{X} and \mathbf{Y} is defined using a set \mathcal{T} of *factor templates*. Each template $T_j \in \mathcal{T}$ defines a set of factors $\{(\mathbf{y}_i, \mathbf{x}_i)\}$, a set K_j of feature indices, parameters $\{\theta_k^j\}_{k \in K_j}$ and feature functions $\{f_k^j\}_{k \in K_j}$. Together they define the following conditional distribution:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_{\mathbf{x}}} \prod_{T_j \in \mathcal{T}} \prod_{(\mathbf{y}_i, \mathbf{x}_i) \in T_j} e^{\sum_{k \in K_j} \theta_k^j f_k^j(\mathbf{y}_i, \mathbf{x}_i)} \quad (3.1)$$

In our case the set \mathcal{T} consists of four templates we will describe below. We construct this graphical model using FACTORIE [McCallum et al., 2009].

Bias Template. We use a bias template T_{Bias} that prefers certain relations *a priori* over others. When the template is unrolled, it creates one factor per variable $Y_{\mathbf{c}}$ for candidate tuple $\mathbf{c} \in C$. The template also consists of one weight θ_r^{Bias} and feature function f_r^{Bias} for each possible relation r . f_r^{Bias} fires if the relation associated with tuple \mathbf{c} is r .

Mention Template. In order to extract relations from text, we need to model the correlation between relation instances and their mentions in text. For this purpose we define the template T_{Men} that connects each relation instance variable $Y_{\mathbf{c}}$ with its observed mention variables $\mathbf{X}_{\mathbf{c}}$. Crucially, this template gathers mentions from multiple documents, and enables us to exploit redundancy.

The feature functions of this template are taken from Mintz et al. [2009]. This includes features that inspect the lexical content between entity mentions in the same

sentence, and the syntactic path between them. One example is

$$f_{101}^{\text{Men}}(y_{\mathbf{c}}, \mathbf{x}_{\mathbf{c}}) \stackrel{\text{def}}{=} \begin{cases} 1 & y_{\mathbf{c}} = \text{founded} \wedge \exists i \text{ with} \\ & \text{“M2 was founded by M1”} \in \mathbf{x}_{\mathbf{c}}^i. \\ 0 & \text{otherwise} \end{cases}$$

It tests whether for any mentions of the candidate tuple the phrase “founded by” appears between the mentions of the argument entities.

Selectional Preferences Templates. To capture the correlations between entity types and relations the entities participate in, we introduce the template T_{Joint} . It connects a relation instance variable Y_{e_1, \dots, e_n} to the individual entity type variables Y_{e_1}, \dots, Y_{e_n} . To measure the compatibility between relation and entity variables, we use one feature $f_{r, t_1 \dots t_a}^{\text{Joint}}$ (and weight $\theta_{r, t_1 \dots t_a}^{\text{Joint}}$) for each combination of relation and entity types $r, t_1 \dots t_a$.

$f_{r, t_1 \dots t_a}^{\text{Joint}}$ fires when the factor variables are in the state $r, t_1 \dots t_a$. For example, $f_{\text{founded}, \text{person}, \text{company}}^{\text{Joint}}$ fires if Y_{e_1} is in state `person`, Y_{e_2} in state `company`, and Y_{e_1, e_2} in state `founded`.

We also add a template T_{Pair} that measures the pairwise compatibility between the relation variable Y_{e_1, \dots, e_a} and each entity variable Y_{e_i} in isolation. Here we use features $f_{i, r, t}^{\text{Pair}}$ that fire if e_i is the i -th argument of \mathbf{c} , has the entity type t and the candidate tuple \mathbf{c} is labelled as instance of relation r . For example, $f_{1, \text{founded}, \text{person}}^{\text{Pair}}$ fires if Y_{e_1} (argument $i = 1$) is in state `person`, and Y_{e_1, e_2} in state `founded`, regardless of the state of Y_{e_2} .

3.3.2 Learning and Inference

Most learning methods need to calculate the model expectations [Lafferty et al., 2001] or the MAP configuration [Collins, 2002] before making an update to the pa-

rameters. This step of inference is usually the bottleneck for learning, even when performed approximately.

SampleRank [Wick et al., 2009] is a rank-based learning framework that alleviates this problem by performing parameter updates *within* MCMC inference. Every pair of consecutive samples in the MCMC chain is ranked according to the model and the ground truth, and the parameters are updated when the rankings disagree. This update can follow different schemes, here we use MIRA [Crammer and Singer, 2003]. This allows the learner to acquire more supervision per instance, and has led to efficient training for models in which inference is expensive and generally intractable [Singh et al., 2009].

There are two types of inference we have to perform: sampling from the posterior during training, and finding the most likely configuration (aka MAP inference). In both settings we employ a Gibbs sampler [Geman and Geman, 1990] that randomly picks a variable Y_c and samples its relation value conditioned on its Markov blanket. At test time we decrease the temperature of our sampler in order to find an approximation of the MAP solution.

3.3.3 Experiments

We set up experiments to answer the following questions: (i) Does the explicit modeling of selectional preferences improve accuracy? (ii) Can we also perform entity and relation extraction in a pipeline and achieve similar results?

We carry out experiments on two data sets, Wikipedia and New York Times articles, and use Freebase as distant supervision source for both. For brevity, I only report the results on New York Times data. Our experimental setting is the same as described in §3.2.2.

For both training and testing we then choose the candidate tuples that may or may not be relation instances. To pick the entities we want to predict entity types

for, we choose all entities that are mentioned at least once in the train/test corpus. To pick the entity pairs that we want to predict the relations of, we choose those that appear at least once together in a sentence.

Since many tuples are not covered in Freebase, for efficiency, we filter out a large fraction of these *negative* candidates for training. The number of negative examples we keep is chosen to be about 10 times the number of positive candidates.

We carry out manual and held-out evaluation. For both evaluation, we rank extracted test relation instances in the MAP state of the network. For manual evaluation we pick the top ranked 50 relation instances for the most frequent relations and ask three annotators to inspect the mentions of these relation instances to decide whether they are correct. Upon disagreement, we use the majority vote. To summarize precisions across relations, we take the average of all relations, and also the average weighted by the proportion of predicted instances for the given relation.

We compare our joint approach against the distant supervision approach [Mintz et al., 2009] and a pipeline approach. For the pipeline approach, we first train an isolated system for entity type prediction. Then we use the output of this system as input for the relation extraction system.

We apply the following configurations of our factor graphs. As our baseline, and roughly equivalent to previous work [Mintz et al., 2009], we pick the templates T_{Bias} and T_{Men} . These describe a fully disconnected graph, and we will refer to this configuration as *isolated*. Next, we add the templates T_{Joint} to model selectional preferences, and refer to this setting as *joint*.

In addition, we evaluate how well selectional preferences can be captured with a simple *pipeline*.

Freebase contains many relation types and only a subset of those relation types occur frequently in the corpus. Since classes with few training instances are generally hard to learn, we restrict ourselves to the 54 most frequently mentioned relations.

These include, for example, `nationality`, `contains`, `founded` and `place_of_birth`. Note that we convert two Freebase non-binary temporal relations to binary relations: `employment_tenure` and `place_lived`. In both cases we simply disregard the temporal information in the Freebase data.

As our main focus is relation extraction, we restrict ourselves to entity types compatible with our selected relations. To this end we inspect the Freebase schema information provided for each relation, and include those entity types that are declared as arguments of our relations. This leads to 10 entity types including `person`, `citytown`, `country`, and `company`.

Note that a Freebase entity can have several types. We pick one of these by choosing the most specific one that is a member of our entity type subset, or `MISC` if no such member exists.

3.3.3.1 Wikipedia data

In our first set of experiments we train and test using Wikipedia as the text corpus. This is a comparatively easy scenario because the facts in Freebase are partly derived from Wikipedia, hence there is an increased chance of properly aligning training facts and text. This is similar to the setting of [Mintz et al. \[2009\]](#).

Held Out Evaluation. We split 1,300,000 Wikipedia articles into training and test sets. Table 3.1 shows the statistics for this split. The last row provides the number of negative relation instances (candidates which are not related according to Freebase) associated with each data set.

Figure 3.6 shows the precision-recall curves of relation extraction for held-out data of various configurations. We notice a slight advantage of the joint approach in the low recall area. Moreover, the joint model predicts more relation instances, as can be seen by its longer line in the graph.

	Wikipedia		NYT	
	Train	Test	Train	Test
#Documents	900K	400K	177K	39K
#Entities	213K	137K	56K	27K
#Positive	36K	24K	5K	2K
#Negative	219K	590K	64K	94K

Table 3.1. The statistics of held-out evaluation on Wikipedia and New York Times.

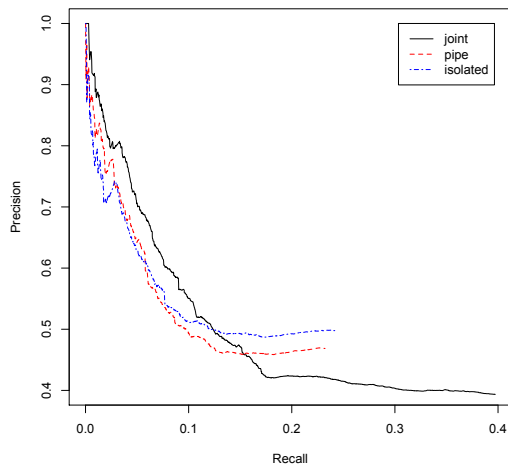


Figure 3.6. Precision-recall curves for various setups in Wikipedia held-out setting.

For higher recall, the joint model performs slightly worse. On closer inspection, we find that this observation is somewhat misleading. Many of the predictions of the joint model are not in the held-out test set derived from Freebase, but nevertheless correct. Hence, to understand if one system really outperforms another, we need to rely on manual evaluation.

Note that the figure only considers binary relations—for entity types all configurations perform similarly.

Manual Evaluation. As mentioned above, held-out evaluation in this context suffers from false negatives in Freebase. Table 3.2 therefore shows the results of our manual evaluation. They are based on the average, and weighted average, of the precisions for the relation instances of the most frequent relations. We notice that

	Isolated	Pipeline	Joint
Wikipedia	0.82	0.87	0.86
Wiki (w)	0.95	0.94	0.95
NYT	0.63	0.65	0.78
NYT (w)	0.78	0.82	0.94

Table 3.2. Average and weighted (w) average precision over frequent relations for New York Times and Wikipedia data, based on manual evaluation.

here all systems perform comparably for weighted average precision. For average precision we see an advantage for both the pipeline and the joint model over the isolated system.

One reason for similar weighted average precisions is the fact that all approaches accurately predict a large number of `contains` instances. This is due to very regular and simple patterns in Wikipedia. For example, most articles on towns start with “A is a municipality in the district of B in C, D.” For these sentences, the relative position of two location mentions is a very good predictor of `contains`. When used as a feature, it leads to high precision for all models. And since `contains` instances are most frequent, and we take the weighted average, results are generally close to each other.

To summarize: in this in-domain setting, modelling compatibility between entity types and relations helps to improve average precision, but not weighted average precision. This holds for both the joint and the pipeline model. However, we will see how this changes substantially when moving to an out-of-domain scenario.

3.3.3.2 New York Times data

We choose all articles of the New York times during 2005 and 2006 as training corpus. As test corpus we use the first 6 months of 2007.

Figure 3.7 shows precision-recall curves for our various setups. We see that jointly modelling entity types and relations helps improve precision.

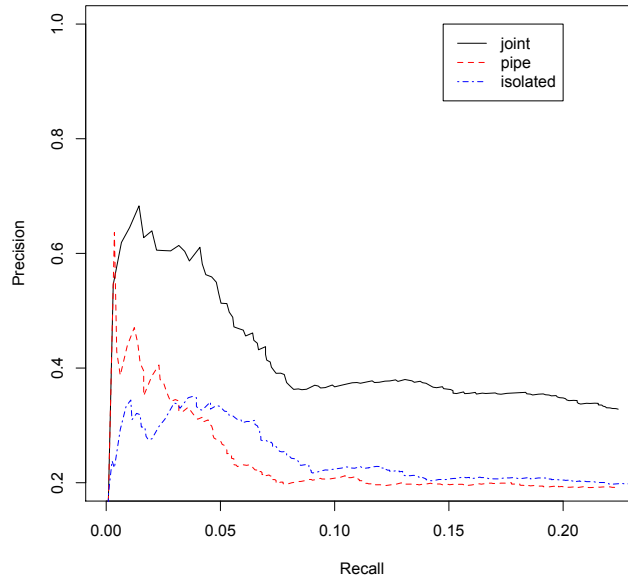


Figure 3.7. Precision-recall curves for isolated, pipeline and joint approaches in New York Times held-out setting.

Due to the smaller overlap between Freebase and NYT data, figure 3.7 also has to be taken with more caution. The systems may predict correct relation instances that just do not appear in Freebase. In this case manual evaluation is even more important.

When evaluating entity precision we find that for both models it is about 84%. This raises the question why the joint entity type and relation extraction model outperforms the pipeline on relations. We take a close look at the entities that participate in relations and find that the joint model performs better on most entity types, for example, `country` and `citytown`. We also look at the relation instances that are predicted by both systems and find that the joint model does predict correct entity types when the pipeline mis-predicts. In fact, exactly these mis-predictions lead the pipeline astray. Considering binary relation instances where the pipeline fails but the joint model does not, we observe an entity precision of 76% for the pipeline and

Relation Type	Iso.	Pipe	Joint
<code>contains</code>	0.92	0.98	0.96
<code>nationality</code>	0.28	0.64	0.82
<code>plc_lived</code>	0.88	0.70	0.96
<code>plc_of_birth</code>	0.32	0.20	0.25
<code>works_for</code>	0.96	0.98	0.98
<code>plc_of_death</code>	0.24	0.40	0.42
<code>children</code>	1.00	0.92	0.98
<code>founded</code>	0.42	0.34	0.71
Avg	0.63	0.65	0.78
Avg(w)	0.78	0.82	0.94

Table 3.3. Precision at 50 for the most frequent relations on New York Times

86% for our joint approach. The joint model fails to correctly predict some entity types that the pipeline gets right, but these tend to appear in contexts where relation instances are easy to extract without considering entity types.²

Manually evaluated precision for New York Times data can be seen in Table 3.3. We can see that modelling entity types and relations jointly makes significant improvement over the baselines. For average precision, our joint model improves over the isolated baseline by 15%, and over the pipeline by 13%. Similar improvements can be observed for weighted average precision.

Let us look at a break-down of precisions with respect to different relations shown in Table 3.3. We see dramatic improvements for `nationality` and `founded` when applying the joint model. Note that the `nationality` relation takes a larger part in the predicted relation instances of the joint model and hence contributes significantly to the weighted average precision.

The algorithm is scalable, and the running time is linear in the number of documents.

²Note that our learned preferences are soft, and hence can be violated in case of wrong entity type predictions.

3.4 Related Work

There are tremendous amounts of work in relation extraction. We briefly review some of them here.

3.4.1 Supervised Relation Extraction

This approach formalizes relation extraction as a multi-class classification task [Zelenko et al., 2003, Culotta and Sorensen, 2004]. Unfortunately it requires large amounts of annotated data and it only extracts relations from one sentence.

3.4.2 Distant Supervision

Learning to extract relations by using distant supervision has raised large amounts of interest in recent years [Craven and Kumlien, 1999, Bunescu and Mooney, 2007, Mintz et al., 2009, Bellare et al., 2007, Bunescu and Mooney, 2007, Weld et al., 2009, Hoffmann et al., 2010]. Both of our works are inspired by Mintz et al. [2009] who also use Freebase as distant supervision source.

Schoenmackers et al. [2008] use entailment rules on assertion extracted by TextRunner to increase recall for relation extraction. They also perform cross-document probabilistic inference based on Markov Networks. However, they do not infer the types of entities and work in an openIE setting.

Many research work has been done to address the mismatching between the distant supervision source and the text data. For example, extending our work, researchers make the assumption that one relation mention can be related by one relation type, can be not related, and one tuple can have multiple relations [Hoffmann et al., 2011, Surdeanu et al., 2012]. Researchers also deal with other challenges in distant supervision to acquire more accurate training data [Min et al., 2013, Ritter et al., 2013].

3.4.3 Joint Entity and Relation Extraction

Joint entity and relation extraction are widely explored in supervised relation and event extraction. For example, [Roth and Yih \[2007\]](#) have used Linear Programming to enforce consistency between entity types and extracted relations. [Kate and Mooney \[2010\]](#) use a pyramid parsing scheme to achieve the same. [Riedel et al. \[2009\]](#) use Markov Logic to model interactions between event-argument relations for biomedical event extraction. These works are in supervised settings, and they perform extraction on a per-sentence basis.

[Carlson et al. \[2010\]](#) also apply the selectional preferences of entity and relation types to improve a bootstrapping process. In each iteration of bootstrapping, extracted facts that violate compatibility constraints will not be used to generate additional patterns in the next iteration.

3.5 Conclusion

In this chapter, we present two models for relation extraction using distant supervision. Our first model relaxes the assumption in distant supervision that every sentence mentioning two related entities expresses the corresponding relation. Instead we use the expressed-at-least-once assumption: at least one (instead of each) sentence which mentions two related entities expresses the corresponding relation. We inject this assumption into a ranking function that is used within SampleRank to discriminatively train a joint model of relation extraction and relation mention prediction. Empirically we show that this approach indeed improves precision: we achieve an error reduction rate of 11.7% compared against the distant supervision baseline.

Our second approach, instead of extracting facts in isolation, models interactions between facts in order to improve precision. In particular, we capture selectional preferences of relations. These preferences are modelled in a cross-document fashion using a large scale factor graph. We show inference and learning can be efficiently

performed in linear time by Gibbs Sampling and SampleRank. When applied to out-of-domain text, this approach leads to a 15% increase in precision over an isolated baseline, and a 13% improvement over a pipelined system.

A crucial aspect of our approach is its extensibility. Since the approach is exclusively framed in terms of an undirected graphical model, it is conceptually easy to extend the model to other types of compatibilities, such as functionality constraints. Our model could also be extended to tackle coreference resolution.

CHAPTER 4

UNSUPERVISED RELATION EXTRACTION USING GENERATIVE MODELS

Distantly supervised relation extraction approach requires a pre-defined knowledge base that contains all entity and relation types that we care about. The available knowledge bases usually cannot fulfill our needs due to their incompleteness. In this chapter, we discuss an unsupervised approach for discovering broad relation and entity types.

4.1 Introduction

Generative models are widely used in discovering latent structures for text documents. We adapt them for relation extraction. We present a series of generative probabilistic models, broadly similar to standard topic models, that generate a corpus of observed triples of entity mention pairs and the surface syntactic dependency path between them. Our proposed models exploit entity type constraints within a relation as well as features on the dependency path between entity mentions. The output of our approach is a clustering over observed relation paths (e.g. “X was born in Y” and “X is from Y”) such that expressions in the same cluster bear the same semantic relation type between entities.

Past work has shown that standard supervised techniques can yield high-performance relation detection when abundant labeled data exists for a fixed inventory of individual relation types (e.g. *placeOfBirth*) [Kambhatla, 2004, Culotta and Sorensen, 2004, Roth and Yih, 2007]. However, less explored are *open-domain* approaches where the

set of possible relation types is not fixed and little to no labeled is given for each relation type [Banko et al., 2007, Banko and Etzioni, 2008]. A more related line of research has explored inducing relation types via clustering. For example, DIRT [Lin and Pantel, 2001] aims to discover different representations of the same semantic relation using distributional similarity of dependency paths. Poon and Domingos [2008] present an Unsupervised semantic parsing (USP) approach to partition dependency trees into meaningful fragments (or “parts” to use their terminology). The combinatorial nature of this dependency partition model makes it difficult for USP to scale to large data sets despite several necessary approximations during learning and inference. Our work is similar to DIRT and USP in that we induce relation types from observed dependency paths, but our approach is a straightforward and principled generative model which can be efficiently learned. As we show empirically, our approach outperforms these related works when trained with the same amount of data and further gains are observed when trained with more data.

We evaluate our approach using ‘intrinsic’ clustering evaluation and ‘extrinsic’ evaluation settings. The former evaluation is performed using subset of induced clusters against Freebase relations, a large manually-built entity and relational database. We also show some clusters which are not included as Freebase relations, as well as some entity clusters found by our approach. The latter evaluation uses the clustering induced by our models as features for relation extraction in distant supervision framework. Empirical results show that we can find coherent clusters. In relation extraction, we can achieve 12% error reduction in precision over a state-of-the-art weakly supervised baseline and we show that using features from our proposed models can find more facts for a relation without significant accuracy loss.

4.2 Models

We present three generative models for modeling tuples of entity mention pairs and the syntactic dependency path between them [Yao et al., 2011]. The first two models, Rel-LDA and Rel-LDA1, are simple extensions of the standard LDA model [Blei et al., 2003]. At the document level, our model is identical to the standard LDA; a multinomial distribution is drawn over a fixed number of relation types R . Changes lie in the observations. In standard LDA, the atomic observation is a word drawn from a latent topic distribution determined by a latent topic indicator variable for that word position. In our approach, a document consists of an exchangeable set of relation tuples. Each relation tuple is drawn from a relation type ‘topic’ distribution selected by a latent relation type indicator variable. Relation tuples are generated using a collection of independent features drawn from the underlying relation type distribution. These changes to standard LDA are intended to have the effect that instead of representing semantically related words, the ‘topic’ latent variable represents a relation type.

Our third model exploits entity type constraints within a relation and induces clusters of relations and entities jointly. For each tuple, a set of relation level features and two latent entity type indicators are drawn independently from the relation type distribution; a collection of entity mention features for each argument is drawn independently from the entity type distribution selected by the entity type indicator.

4.2.1 Rel-LDA Model

This model is an extension to the standard LDA model. At the document level, a multinomial distribution over relations θ_{doc} is drawn from a prior $\text{Dir}(\alpha)$. To generate a relation tuple, we first draw a relation ‘topic’ r from $\text{Multi}(\theta)$. Then we generate each feature f of a tuple independently from a multinomial distribution $\text{Multi}(\phi_{rf})$ selected by r . In this model, each tuple has three features, i.e. its three components,

Path	X, made by Y
Source	Gamma Knife
Dest	Elekta
Trigger	make
Lex	, made by the Swedish medical technology firm
POS	, VBN IN DT JJ JJ NN NN
NER pair	MISC-ORG

Table 4.1. The features of tuple ‘(Gamma Knife, made by, Elekta)’ in sentence “Gamma Knife, made by the Swedish medical technology firm Elekta, focuses low-dosage gamma radiation ...”

shown in the first three rows in Table 4.1. Figure 4.1 shows the graphical representation of Rel-LDA. Table 4.2 lists all the notation used in describing our models.

$ R $	Number of relations
$ D $	Number of documents
r	A relation
doc	A document
p, s, d	Dep path, source and dest args
f	A feature/feature type
T	Entity type of one argument
α	Dirichlet prior for θ_{doc}
β_x	Dirichlet prior for ϕ_{rx}
β	Dirichlet prior for ϕ_t
θ_{doc}	$p(r doc)$
ϕ_{rx}	$p(x r)$
ϕ_t	$p(f_s T), p(f_d T)$

Table 4.2. The notation used in our models

We employ Gibbs sampling to learn the parameters. It is an expectation maximization (EM) approach. The procedure is similar to that used by the standard topic model. In the E-step (inference), we sample the relation type indicator for each tuple using $p(r|f)$:

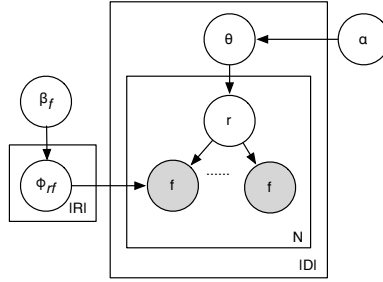


Figure 4.1. Rel-LDA model. Shaded circles are observations, and unshaded ones are hidden variables. A document consists of N tuples. Each tuple has a set of features. Each feature of a tuple is generated independently from a hidden relation variable r .

$$\begin{aligned}
 P(r|f(p, s, d)) &\propto p(r) \prod_f p(f|r) \\
 &\propto (\alpha_r + n_{r|d}) \prod_f \frac{\beta_f + n_{f|r}}{\sum_{f'} (\beta_{f'} + n_{f'|r})}
 \end{aligned}$$

We estimate $p(r)$ and $p(f|r)$ in the M-step:

$$\begin{aligned}
 \theta_{doc} &= \frac{\alpha + n_{r|doc}}{\sum_{r'} (\alpha + n_{r'|doc})} \\
 \phi_{rf} &= \frac{\beta_f + n_{f|r}}{\sum_{f'} (\beta_{f'} + n_{f'|r})}
 \end{aligned}$$

where $n_{f|r}$ indicates the number of times a feature f is assigned with r .

4.2.2 Rel-LDA1 model

Looking at results of Rel-LDA, we find the clusters sometimes are in need of refinement, and we can address this by adding more features. For instance, adding *trigger* features can encourage sparsity over dependency paths. We define trigger words as all the words on the dependency path except stop words. For example, from path “X, based in Y,” “base” is extracted as a trigger word. The intuition for using trigger words is that paths sharing the same set of trigger words should go to one cluster. Adding named entity tag pairs can refine the clusters too. For example, a cluster produced by Rel-LDA contains “X was born in Y” and “X lives in Y;” but it

also contains “X, a company in Y.” In this scenario, adding features ‘PER-LOC’ and ‘ORG-LOC’ can push the model to split the clusters into two and put the third case into a new cluster.

Hence we propose Rel-LDA1. It is similar to Rel-LDA, except that each tuple is represented with more features. Besides p , s , and d , we introduce trigger words, lexical pattern, POS tag pattern, and the named entity pair features for each tuple. Lexical pattern is the word sequence between the two arguments of a tuple and POS tag pattern is the POS tag sequence of the lexical pattern. See Table 4.1 as an example.

Following typical EM learning [Charniak and Elsnar, 2009], we start with a much simpler generative model, expose the model to fewer features first, and iteratively add more features. First, we train a Rel-LDA model, i.e. the model only generates the dependency path, source and destination arguments. After each interval of 10 iterations, we introduce one additional feature. We add the features in the order of trigger, lexical pattern, POS, and NER pair.

4.2.3 Type-LDA model

We know that relations can only hold between certain entity types, known as selectional preferences [Ritter et al., 2010, Seaghdha, 2010, Kozareva and Hovy, 2010]. We propose a Type-LDA model that can capture the selectional preferences of relations to their arguments. This model clusters tuples into relational clusters, and arguments into different entity clusters. The entity clusters could be useful in many applications, for example, defining fine-grained entity types and finding new concepts.

We split the features of a tuple into relation level features and entity level features. Relation level features include the dependency path, trigger, lex and POS features; entity level features include the entity mention itself and its named entity tag.

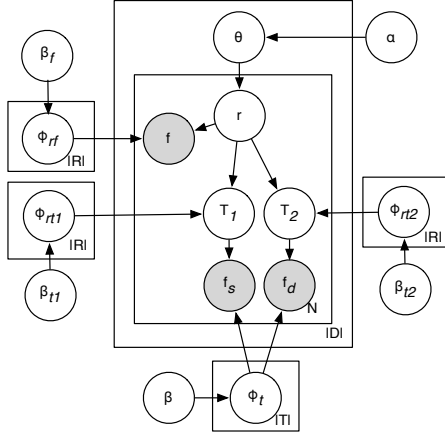


Figure 4.2. Type-LDA model. Each document consists of N tuples. Each tuple has a set of features, relation level features f and entity level features of source argument f_s and destination argument f_d . Relation level features and two hidden entity types T_1 and T_2 are generated from hidden relation variable r independently. Source entity features are generated from T_1 and destination features are generated from T_2 .

The generative storyline is as follows. At the document level, the model draws a multinomial distribution over relations θ_{doc} from a Dirichlet prior. A document consists of N relation tuples. Each tuple is represented by relation level features (f) and entity level features of source argument (f_s) and destination argument (f_d). The model draws a relation r from $\text{Multi}(\theta_{doc})$ for each tuple. Subsequently, the model generates independently the relation level features and two hidden entity types T_1 and T_2 from r . Finally, the model generates features f_s from T_1 and f_d from T_2 . Figure 4.2 shows the graphical representation of this model.

At inference time, we sample r , T_1 and T_2 for each tuple. For efficient inference, we first initialize the model without T_1 and T_2 , i.e. generating all the features directly from r . Here the model degenerates to Rel-LDA1. After some iterations, we introduce T_1 and T_2 . We sample the relation variable (r) and two mention types variables (T_1, T_2) iteratively for each tuple. We can sample them together, but this is not efficient. In addition, we found that it does not improve performance.

4.3 Experiments

As mentioned in §2.2, our data is a list of tuples. A tuple has two entity mentions and a dependency path between them. Following DIRT [Lin and Pantel, 2001], we filter out tuples that do not satisfy the following constraints. First, the path needs to be shorter than 10 edges, since longer paths occur less frequently. Second, the dependency relations in the path should connect two content words, i.e. nouns, verbs, adjectives and adverbs. For example, in phrase “solve a problem,” the dependency relation “obj(solve, problem)” is kept, while “det(problem, a)” is discarded. Finally, the dependency labels on the path must not fall into the set including ‘conj,’ ‘ccomp,’ ‘parataxis,’ ‘xcomp,’ and so on. This selection is based on the observation that most of the times the corresponding dependency relations do not explicitly state a relation between two candidate arguments.

After all entity mentions are generated and paths are extracted, we are left with nearly 2.5M tuples. After clustering (inference), each of these tuples belongs to one cluster/relation and is associated with its clusterID.

We experimented with the number of clusters and find that in the range of 50-200 the performance does not vary significantly with different numbers. In our experiments, we cluster the tuples into 100 relation clusters for all three models. For Type-LDA model, we use 50 entity clusters.

We evaluate our models in both ‘internal’ and ‘external’ ways. The internal evaluation aims at measuring the clustering quality by mapping clusters to Freebase relations. The external one seeks to assess the utility of our predicted clusters as features for relation extraction.

4.3.1 Relations discovered by different models

Looking closely at the clusters we predict, we find that some of them can be mapped to Freebase relations. We discover clusters that roughly correspond to the

parentCom (parent company relation), *filmDirector*, *authorOf*, *comBase* (base of a company relation) and *dieIn* relations in Freebase. We treat Freebase annotations as ground truth and measure recall. We count each tuple in a cluster as true positive if Freebase states the corresponding relation between its argument pair. We find that precision numbers against Freebase are low, below 10%. However, these numbers are not reliable mainly because many correct instances found by our models are missing in Freebase. One reason why our predictions are missing in Freebase is coreference. For example, we predict *parentCom* relation between “Linksys” and “Cisco,” while Freebase only considers “Cisco Systems, Inc.” as the parent company of “Linksys.” It does not corefer “Cisco” to “Cisco Systems, Inc.” Incorporating coreference in our model may fix this problem. Instead of measuring precision against Freebase, we ask humans to label 50 instances for each cluster and report precision according to this annotated data. Table 4.3 shows the scores.

Rel.	Sys.	Rec.	Prec.
parentCom	Rel-LDA	51.4	76.0
	Rel-LDA1	49.5	78.0
	Type-LDA	55.3	72.0
filmDirector	Rel-LDA	42.5	32.0
	Rel-LDA1	70.5	40.0
	Type-LDA	74.2	26.0
comBase	Rel-LDA	31.5	12.0
	Rel-LDA1	54.2	22.0
	Type-LDA	57.1	30.0
authorOf	Rel-LDA	25.2	84.0
	Rel-LDA1	46.9	86.0
	Type-LDA	20.2	68.0
dieIn	Rel-LDA	26.5	34.0
	Rel-LDA1	55.9	40.0
	Type-LDA	50.2	28.0

Table 4.3. Clustering quality evaluation (%). Recall is measured against Freebase. Precision is measured according to human annotators

We can see that in most cases Rel-LDA1 and Type-LDA substantially outperform the Rel-LDA model. This is due to the fact that both models can exploit more features to make clustering decisions. For example, in Rel-LDA1 model, the NER pair feature restricts the entity types the two arguments can take.

In the following, we analyze the behaviors of different models by examples. Considering *parentCom* relation, Rel-LDA includes spurious instances such as “A is the chief executive of B,” whereas Rel-LDA1 has fewer such instances due to the NER pair feature. Similarly, by explicitly modeling entity type constraints, Type-LDA makes fewer such errors. All our models make mistakes when sentences have coordination structures on which the parser has failed. For example, when a sentence has the following pattern “The winners are A, a part of B; C, a part of D; E, a part of F,” our models may predict *parentCom*(A,F), because the parser connects A with F via the pattern “a part of.”

Some clusters found by our models cannot be mapped to Freebase relations. Take the Freebase relation *worksFor* as one example. This relation subsumes all types of employment relationships, irrespective of the role the employee plays for the employer. By contrast, our models discover clusters such as *leaderOf*, *editorOf* that correspond to more specific roles an employee can have. We show some example relations in Table 4.4. In the table, the 2nd row shows a cluster of employees of news media companies; the 3rd row shows leaders of companies; the last one shows birth and death places of persons. We can see that the last cluster is noisy since we do not handle antonyms in our models. The arguments of the clusters are noisy too. For example, ‘New York’ occurs as a destination argument in the 2nd cluster. This is because ‘New York’ has high frequency in the corpus and it brings noise to the clustering results. In Table 4.5 we show some entity clusters produced by Type-LDA. We find different types of companies, such as financial companies and news companies. We also find subclasses of *person*, for example, *reviewer* and *politician*, because these different entity classes

Source	New York, Euro RSCG Worldwide, BBDO Worldwide, American, DDB Worldwide
Path	X, a part of Y; X, a unit of Y; X unit of Y; X, a division of Y; X is a part of Y
Dest	Omnicom Group, Interpublic Group of Companies, WPP Group, Publicis Groupe
Source	Supreme Court, Anna Wintour, William Kristol, Bill Keller, Charles McGrath
Path	X, an editor of Y; X, a publisher of Y; X, an editor in chief of Y;
Dest	The Times, The New York Times, Vogue, Vanity Fair, New York
Source	Kenneth L. Lay, L. Dennis Kozlowski, Bernard J. Ebbers, Thomas R. Suozzi, Bill Gates
Path	X, the executive of Y; X, Y executive; X, the chairman of Y
Dest	Enron, Microsoft, WorldCom, Citigroup, Nassau County
Source	Paul J. Browne, John McArdle, Tom Cocola, Claire Buchan, Steve Schmidt
Path	X, a spokesman for Y; X, a spokeswoman for Y; X, a commissioner of Y
Dest	White House, Justice Department, Pentagon, United States, State Department
Source	United Nations, Microsoft, Intel, Internet, M. D. Anderson
Path	X, based in Y; X, which is based in Y; X, a company in Y; X, a consultant in Y
Dest	New York, Washington, Manhattan, Chicago, London
Source	Army, Shiite, Navy, John, David
Path	X was born in Y; X die at home in Y; X, son of Y; X die at Y
Dest	Manhattan, World War II, Brooklyn, Los Angeles, New York

Table 4.4. The path, source and destination arguments of some relations found by Rel-LDA1.

participate in different relations. The last cluster shown in the table is a mixture of news companies and government agencies. This may be because this entity cluster is affected by many relations.

Company	Microsoft, Enron, NBC, CBS, Disney
FinanceCom	Merrill Lynch, Morgan Stanley, Goldman Sachs, Lehman Brothers
News	Notebook, New Yorker, Vogue, Vanity Fair, Newsweek
SportsTeam	Yankees, Mets, Giants, Knicks, Jets
University	University of California, Harvard, Columbia University
Art Reviewer	Stephen Holden, Ken Johnson, Roberta Smith, Anthony Tommasini
Games	World Series, Olympic, World Cup, Super Bowl, Olympics
Politician	Eliot Spitzer, Ari Fleischer, Kofi Annan, Scott McClellan, Karl Rove
Gov. Agency	Congress, European Union, NATO, Federal Reserve
News/Agency	The New York Times, The Times, Supreme Court, Security Council

Table 4.5. The entity clusters found by Type-LDA

4.3.2 Distant Supervision based Relation Extraction

Our generative models detect clusters of dependency paths and their arguments. Such clusters are useful in their own right, but we claim that they can also help a

supervised relation extractor. We validate this hypothesis in the context of relation extraction with distant supervision using predicted clusters as features.

Following previous work [Mintz et al., 2009], we use Freebase as our distant supervision source, and align related entity pairs to the New York Times articles discussed earlier. Our training and test instances are pairs of entities for which both arguments appear in at least one sentence together. Features of each instance are extracted from all sentences in which both entities appear together. The gold label for each instance comes from Freebase. If a pair of entities is not related according to Freebase, we consider it a negative example. Note that this tends to create some amount of noise: some pairs may be related, but their relationships are not yet covered in Freebase.

After filtering out relations with fewer than 10 instances we have 65 relations and an additional “O” label for unrelated pairs of entities. We call related instances positive examples and unrelated instances negative examples.

We train supervised classifiers using maximum entropy. The baseline classifier employs features that Mintz et al. [2009] used. To extract features from the generative models we proceed as follows. For each pair of entities, we collect all tuples associated with it. For each of these tuples we extract its clusterID, and use this ID as a binary feature.

The baseline system without generative model features is called *Distant*. The classifiers with additional features from generative models are named after the generative models. Thus we have Rel-LDA, Rel-LDA1 and Type-LDA classifiers. We compare these against Distant and the DIRT database. For the latter we parse our data using Minipar [Lin, 1998] and extract dependency paths between pairs of named entity mentions. For each path, the top 3 similar paths are extracted from DIRT database. The Minipar path and the similar paths are used as additional features.

For held-out evaluation, we construct the training data from half of the positive examples and half of the negative examples. The remaining examples are used as

test data. Note that the number of negative instances is more than 10 times larger than the number of positive instances. At test time, we rank the predictions by the conditional probabilities obtained from the Maximum Entropy classifier. We report precision of top ranked 50 instances for each relation in table 4.6. From the table we can see that all systems using additional features outperform the Distant system. In average, our best model achieves 4.1% improvement over the distant supervision baseline, 12% error reduction. The precision of *bornIn* is low because in most cases we predict *bornIn* instances as *liveIn*.

We expect systems using generative model features to have higher recall than the baseline. This is difficult to measure, but precision in the high recall area is a signal. We look at top ranked 1000 instances of each system and show the precision in the last row of the table. We can see that our best model Type-LDA outperforms the distant supervision baseline by 4.5%.

Relation	Dist	Rel	Rel1	Type	DIRT
worksFor	80.0	92.0	86.0	90.0	84.0
authorOf	98.0	98.0	98.0	98.0	98.0
containedBy	92.0	96.0	96.0	92.0	96.0
bornIn	16.0	18.0	22.0	24.0	10.0
dieIn	28.0	30.0	28.0	24.0	24.0
liveIn	50.0	52.0	54.0	54.0	56.0
nationality	92.0	94.0	90.0	90.0	94.0
parentCom	94.0	96.0	96.0	96.0	90.0
founder	65.2	76.3	61.2	64.0	68.3
parent	52.0	54.0	50.0	52.0	52.0
filmDirector	54.0	60.0	60.0	64.0	62.0
Avg	65.6	69.7	67.4	68.0	66.8
Prec@1K	82.8	85.8	85.3	87.3	82.8

Table 4.6. Precision (%) of some frequent relations

Why do generative model features help in improving relation extraction? One reason is that generative models can transfer information from known patterns to unseen patterns. For example, given “Sidney Mintz, the great food anthropologist at

Johns Hopkins University,” we want to predict the relation between “Sidney Mintz” and “Johns Hopkins University.” The distant supervision system incorrectly predicts the pair as ‘O’ since it has not seen the path “X, the anthropologist at Y” in the training data. By contrast, Rel-LDA can predict this pair correctly as *worksFor*, since the dependency path of this pair is in a cluster which contains the path “X, a professor at Y,” and this path in turn is a strong indicator for relation *worksFor*.

In addition to held-out evaluation we also carry out manual evaluation. To this end, we use all the positive examples and randomly select five times the number of positive examples as negative examples to train a classifier. The remaining negative examples are candidate instances. We rank the predicted instances according to their classification scores. For each relation, we ask human annotators to judge its top ranked 50 instances.

Table 4.7 lists the manual evaluation results for some frequent relations. We also list how many instances are extracted for each relation. For almost all the relations, systems using generative model features find more instances. In terms of precision, our models perform comparatively to the baseline, even better for some relations.

Relation	Top 50 (%)			#Instances		
	Distant	RelLDA	TypeLDA	Distant	RelLDA	TypeLDA
worksFor	100.0	100.0	100.0	314	349	349
authorOf	94.0	94.0	96.0	185	208	229
containedBy	98.0	98.0	98.0	670	714	804
bornIn	82.6	88.2	88.0	46	36	56
dieIn	100.0	100.0	100.0	167	176	231
liveIn	98.0	98.0	94.0	77	86	109
nationality	78.0	82.0	76.0	84	92	114
parentCom	79.2	77.4	85.7	24	31	28
founder	80.0	80.0	50.0	5	5	14
parent	97.0	92.3	94.7	33	39	38
filmDirector	92.6	96.9	97.1	27	32	34

Table 4.7. Manual evaluation, Precision and recall of some frequent relations

We also notice that clustering quality is not consistent with distant supervision performance. Rel-LDA1 can find better clusters than Rel-LDA but it has lower precision in held-out evaluation. Type-LDA underperforms Rel-LDA in average precision but it gets higher precision in a higher recall area, i.e. precision at 1K. One possible reason for the inconsistency is that the baseline distant supervision system already employs features that are used in Rel-LDA1. Another reason may be that the clusters do not overlap with Freebase relations well, see §4.3.1.

4.3.3 Comparing against USP

We also compare against USP [Poon and Domingos, 2008]. Due to memory requirements of USP, we are only able to run it on a smaller data set consisting of 1,000 NYT documents; this is three times the amount of data Poon and Domingos [2008] used to train USP.¹ For distant supervision based relation extraction, we only match approximately 500 Freebase instances to this small data set.

USP provides a parse tree for each sentence and we can extract a path from the tree for each mention pair. Since USP provides clusters of words and phrases, we use the USP clusterID associated with the words on the path as binary features in the classifier.

All models are less accurate when trained on this smaller dataset; we can do as well as USP does, even a little better. USP achieves 8.6% in F1, Rel-LDA 8.7%, Rel-LDA1 10.3%, Type-LDA 8.9% and Distant 10.3%. Of course, given larger datasets, the performance of Rel-LDA, Rel-LDA1, and Type-LDA improves considerably. In summary, comparing against USP, our approach scales much more easily to large data.

¹Using the publicly released USP code, training a model with 1,000 documents resulted in about 45 gigabytes of heap space in the JVM.

4.4 Related Work

Supervised and weakly supervised approaches for relation extraction cannot discover new relations and classify instances which do not belong to any of the predefined relations. Researchers have devoted large amounts of efforts on inducing relations using unsupervised approaches. We briefly list several examples here.

DIRT [Lin and Pantel, 2001] aims to discover different representations of the same semantic relation, i.e. similar dependency paths. They employ the distributional similarity based approach while we use generative models. Both DIRT and our approach take advantage of the arguments of dependency paths to find semantic relations. Moreover, our approach can cluster the arguments into different types.

Unsupervised semantic parsing (USP) [Poon and Domingos, 2008] discovers relations by merging predicates which have similar meanings; it proceeds to recursively cluster dependency tree fragments (or “parts”) to best explain the observed sentence. It does not concentrate on capturing any particular kind of relation between sentence constituents, but on capturing repeated patterns. Our approach differs in that we aim to capture a narrow range of binary relations between named entities; some of our models (see § 4.2) explore entity type information to constraint relation type induction. Also, our models are scalable and we train them on a large corpus. In addition, we use a distant supervision framework for evaluation.

Relation duality [Bollegala et al., 2010] employs co-clustering to find clusters of entity pairs and patterns. They identify each cluster of entity pairs as a relation by selecting representative patterns for that relation. This approach is related to our models, however, it does not identify any entity clusters.

Generative probabilistic models are widely employed in relation extraction. For example, they are used for in-domain relation discovery while incorporating constraints via posterior regularization [Chen et al., 2011]. We are focusing on open domain relation discovery. Generative models are also applied to selectional pref-

erence discovery [Ritter et al., 2010, Seaghdha, 2010]. In this scenario, the authors assume relation labels are given while we automatically discover relations. Generative models are also used in unsupervised coreference [Haghighi and Klein, 2010].

Clustering is also employed in relation extraction. Hasegawa et al. [2004] cluster pairs of named entities according to the similarity of context words intervening between them. Their approach is not probabilistic. Researchers also use topic models to perform dimension reduction on features when they cluster relations [Hachey, 2009]. However, they do not explicitly model entity types.

Open information extraction aims to discover relations independent of specific domains and relations [Banko et al., 2007, Banko and Etzioni, 2008]. A self-learner is employed to extract relation instances but the systems do not cluster the instances into relations. Yates and Etzioni [2009] present RESOLVER for discovering relational synonyms as a post processing step. Our approach integrates entity and relation discovery in a probabilistic model.

4.5 Conclusion

In this chapter, we present an unsupervised probabilistic generative approach to relation extraction between two named entities. Our proposed models explore entity type constraints within a relation as well as features on the dependency path between entity mentions to cluster equivalent textual expressions. We demonstrate the effectiveness of this approach by comparing induced relation clusters against a large knowledge base. We also show that using clusters of our models as features in distant supervised framework yields 12% error reduction in precision over a weakly supervised baseline and outperforms other state-of-the art relation extraction techniques.

CHAPTER 5

UNSUPERVISED RELATION DISCOVERY WITH SENSE DISAMBIGUATION

To discover relation types from text, most methods cluster shallow or syntactic patterns of relation mentions, but consider only one possible sense per pattern [Bollegala et al., 2010, Lin and Pantel, 2001, Yao et al., 2011]. In practice this assumption is often violated. In this chapter we present an approach to overcome this issue by inducing clusters of pattern senses from feature representations of patterns. In particular, we employ a topic model to partition entity pairs associated with patterns into sense clusters using local and global features. We merge these sense clusters into semantic relations using hierarchical agglomerative clustering. We compare against several baselines: a generative latent-variable model, a clustering method that does not disambiguate between path senses, and our own approach but with only local features. Experimental results show our proposed approach discovers dramatically more accurate clusters than models without sense disambiguation, and it is crucial to incorporate global features, such as the document theme.

5.1 Introduction

Many relation discovery methods rely exclusively on the notion of either shallow or syntactic patterns that appear between two named entities [Bollegala et al., 2010, Lin and Pantel, 2001]. Such patterns could be sequences of lemmas and POS tags, or lexicalized dependency paths. Generally speaking, relation discovery attempts to cluster such patterns into sets of equivalent or similar meaning. Whether we use sequences or

dependency paths, we will encounter the problem of polysemy. For example, a pattern such as “A beat B” can mean that person A wins over B in competing for a political position, as pair “(Hillary Rodham Clinton, Jonathan Tasini)” in “Sen Hillary Rodham Clinton beats rival Jonathan Tasini for Senate.” It can also indicate that an athlete A beat B in a sports match, as pair “(Dmitry Tursunov, Andy Roddick)” in “Dmitry Tursunov beat the best American player Andy Roddick.” Moreover, it can mean “physically beat” as pair “(Mr. Harris, Mr. Simon)” in “On Sept. 7, 1999, Mr. Harris fatally beat Mr. Simon.” This is known as *polysemy*. If we work with patterns alone, our extractor will not be able to differentiate between these cases.

A large body of previous work does not explicitly address this problem. [Lin and Pantel \[2001\]](#) assumes only one sense per path. [Pantel et al. \[2007\]](#) augment each relation with its selectional preferences, i.e. fine-grained entity types of two arguments, to handle polysemy. However, such fine grained entity types come at a high cost. It is difficult to discover a high-quality set of fine-grained entity types due to unknown criteria for developing such a set. In particular, the optimal granularity of entity types depends on the particular pattern we consider. For example, a pattern like “A beat B” could refer to A winning a sports competition against B, or a political election. To differentiate between these senses we need types such as “Politician” or “Athlete.” However, for “A, the parent of B” we only need to distinguish between persons and organizations (for the case of the sub-organization relation). In addition, there are senses that just cannot be determined by entity types alone, like our example “A beat B,” the entity type “person” for A and B could not disambiguate the senses.

In this paper we address the problem of polysemy, while we circumvent the problem of finding fine-grained entity types. Instead of mapping entities to fine-grained types, we directly induce pattern senses by clustering feature representations of pattern contexts, i.e. the entity pairs associated with a pattern. This allows us to employ

not only local features such as words, but also global features such as the document and sentence themes.

To cluster the entity pairs of a single relation pattern into senses, we develop a simple extension to Latent Dirichlet Allocation [Blei et al., 2003]. Once we have our pattern senses, we merge them into clusters of different patterns so that patterns in the same cluster have similar sense. We employ hierarchical agglomerative clustering with a similarity metric that considers features such as the entity arguments, and the document and sentence themes.

We perform experiments on New York Times articles and consider lexicalized dependency paths as patterns in our data. In the following we shall use the term path and pattern exchangeably. We compare our approach with several baseline systems, including a generative model approach, a clustering method that does not disambiguate between senses, and our approach with different features. We perform both automatic and manual evaluations. For automatic evaluation, we use relation instances in Freebase as ground truth, and employ two clustering metrics, pairwise F-score and B^3 (as used in coference). Experimental results show that our approach improves over the baselines, and that using global features achieves better performance than using entity type based features. For manual evaluation, we employ a set intrusion method [Chang et al., 2009]. The results also show that our approach discovers relation clusters that human evaluators find coherent.

5.2 Our Approach

We induce pattern senses by clustering the entity pairs associated with a pattern, and discover semantic relations by clustering these sense clusters [Yao et al., 2012b]. We represent each pattern as a list of entity pairs and employ a topic model to partition them into different sense clusters using local and global features. We take each sense cluster of a pattern as an atomic cluster, and use hierarchical agglomerative

clustering to organize them into semantic relations. Therefore, a semantic relation comprises a set of sense clusters of patterns. Note that one pattern can fall into different semantic relations when it has multiple senses.

5.2.1 Sense Disambiguation

In this section, we discuss the details of how we discover senses of a pattern. We form a clustering task by collecting all entity pairs a pattern connects. Our goal is to partition these entity pairs into sense clusters. After this clustering, each pattern will have multiple senses. Each sense is represented by a cluster of entity pairs. We represent each entity pair by the following features.

Entity names: Since participating arguments can differentiate pattern senses, we use the surface string of an entity pair as features. For example, intuitively, for one pattern “A play B,” pairs which contain the B argument “Mozart” and pairs which have the B argument “Mets” (an american football team) should be in different sense clusters.

Words: The words between and around the two entity arguments can disambiguate the sense of a path. For example, “A’s parent company B” is different from “A’s largest company B” although they share the same path “A’s company B.” The former describes the sub-organization relationship between two companies, whereas the latter describes B as the largest company in a location A. The two words to the left of the source argument, and to the right of the destination argument also help sense discovery. For example, in “Mazurkas played by Anna Kijanowska, pianist,” “pianist” tells us pattern “A played by B” takes the “music” sense.

Document theme: Sometimes, the same pattern can express different relations in different documents, depending on the document’s theme. For instance, in a document about politics, “A defeated B” is perhaps about a politician that won an election against another politician. While in a document about sports, it could be a team that

won against another team in a game, or an athlete that defeated another athlete. In our experiments, we use the meta-descriptors of a document as side information and train a standard LDA model to find the theme of a document. See §5.3.1 for details.

Sentence theme: A document may cover several themes. Moreover, sometimes the theme of a document is too general to disambiguate senses. We therefore also extract the theme of a sentence as one feature. Details are in §5.3.1.

We call entity name and word features as local, and the two theme features as global.

We employ a topic model to discover senses for each path based on the feature representations of entity pairs. Each path p_i forms a document, and it contains a list of entity pairs co-occurring with the path. Each entity pair is represented by a list of features f_k . For each path, we draw a multinomial distribution θ over topics/senses. For each feature of an entity pair, we draw a topic/sense from θ_{p_i} . Formally, the generative process is as follows:

$$\begin{aligned}\theta_{p_i} &\sim \text{Dirichlet}(\alpha) \\ \phi_z &\sim \text{Dirichlet}(\beta) \\ z_e &\sim \text{Multinomial}(\theta_{p_i}) \\ f_k &\sim \text{Multinomial}(\phi_{z_e})\end{aligned}$$

Assume we have m paths and l entity pairs for each path. We denote each entity pair of a path as $e(p_i) = (f_1, \dots, f_n)$. Hence we have:

$$\begin{aligned}P(e_1(p_i), e_2(p_i), \dots, e_l(p_i) | z_1, z_2, \dots, z_l) \\ = \prod_{j=1}^l \prod_{k=1}^n p(f_k | z_j) p(z_j)\end{aligned}$$

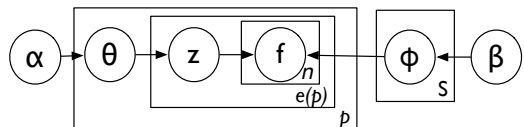


Figure 5.1. Sense-LDA model.

We assume the features are conditionally independent given the topic assignments. Each feature is generated from a multinomial distribution ϕ . We apply Dirichlet priors on θ and ϕ . Figure 5.1 shows the graphical representation of this model.

This model is a minor variation on the standard LDA model. The difference is that instead of drawing an observation from a hidden topic variable, we draw multiple observations from a hidden topic variable. Gibbs sampling is used for inference. After inference, each entity pair of a path is assigned to one topic. One topic is one sense. Entity pairs which share the same topic assignments form one sense cluster.

5.2.2 Hierarchical Agglomerative Clustering

After discovering sense clusters of paths, we employ hierarchical agglomerative clustering (HAC) to discover semantic relations from these sense clusters. We apply the complete linkage strategy and take cosine similarity as the distance function. The cutting threshold is set to 0.1.

We represent each sense cluster as one vector by summing up features from each entity pair in the cluster. The weight of a feature indicates how many entity pairs in the cluster have the feature. Some features may get larger weights and dominate the cosine similarity. We down-weight these features. For example, we use binary features for word “defeat” in sense clusters of pattern “A defeat B”. The two theme features are extracted from generative models, and each is a topic number.

Our approach produces sense clusters for each path and semantic relation clusters of the whole data. Table 5.1 and 5.2 show some example output.

20:sports	30:entertainment	25:music/art
Americans, Ireland Yankees, Angels Ecuador, England Redskins, Detroit Red Bulls, F.C. Barcelona	Jean-Pierre Bacri, Jacques Rita Benton, Gay Head Dance Jeanie, Scrabble Meryl Streep, Leilah Kevin Kline, Douglas Fairbanks	Daniel Barenboim, Mozart Mr. Rose, Ballade Gil Shaham, Violin Romance Ms. Golabek, Steinways Bruce Springsteen, Saints
sports game yankees beat victory num-num won -	music books television theater production book film show played plays directed artistic r:theater	music theater music reviews opera director conducted production r:theater r:hall r:york l:opera

Table 5.1. Example sense clusters for pattern “A play B” produced by sense disambiguation. For each sense, we randomly sample 5 entity pairs. We also show top features for each sense. Each row shows one feature type, where “num” stands for digital numbers, and prefix “l:” for source argument, prefix “r:” for destination argument. Some features overlap with each other. We manually label each sense for easy understanding. We can see the last two senses are close to each other. For two theme features, we replace the theme number with the top words. For example, the document theme of the first sense is Topic30, and Topic30 has top words “sports”. The lower four rows are four types of features: document theme, sentence theme, lexical words and entity names.

5.3 Experiments

We carry out experiments on New York Times articles. Our data is a list of triples. We filter out paths which occur fewer than 200 times and use some heuristic rules to filter out paths which are unlikely to represent a relation, for example, paths in with both arguments take the syntactic role “dobj” (direct objective) in the dependency path. In such cases both arguments are often part of a coordination structure, and it is unlikely that they are related. In summary, we collect about one million tuples, 1300 patterns and half million named entities. In terms of named entities, the data is very sparse. On average one named entity occurs four times.

5.3.1 Feature Extraction

For the entity name features, we split each entity string of a tuple into tokens. Each token is a feature. The source argument tokens are augmented with prefix “l:”,

relation	paths
entertainment	A, who play B:30; A play B:30; star A as B:30
sports	lead A to victory over B:20; A play B:20; A's loss to B:20; A trail B:20; A face B:26; A hold B:26; A play B:26; A acquire (X) from B:26
politics	A nominate B:39; A name B:39; A select B:39; A select B:42; A ask B:42; A choose B:42; A nominate B:42; A turn to B:42
law	A charge B:39; A file against B:39; A accuse B:39; A sue B:39

Table 5.2. Example semantic relation clusters produced by our approach. For each cluster, we list the top paths in it, and each is followed by “:number”, indicating its sense obtained from sense disambiguation. They are ranked by the number of entity pairs they take. The column on the left shows sense of each relation. They are added manually by looking at the sense numbers associated with each path.

and the destination argument tokens with prefix “r:”. We use tokens to encourage overlap between different entities.

For the word features, we extract all the words between the two arguments, removing stopwords and the words with capital letters. Words with capital letters are usually named entities, and they do not tend to indicate relations. We also extract neighboring words of source and destination arguments. The two words to the left of the source argument are added with prefix “lc:” Similarly the two words to the right of the destination arguments are added with prefix “rc:”

Each document in the NYT corpus is associated with many descriptors, indicating the topic of the document. For example, some documents are labeled as “Sports,” “Dallas Cowboys,” “New York Giants,” “Pro Football” and so on. Some are labeled as “Politics and Government,” and “Elections.” We extract a theme feature for each document from these descriptors. To this end we interpret the descriptors as words in documents, and train a standard LDA model based on these documents. We pick the most frequent topic as the theme of a document.

We also train a standard LDA model to obtain the theme of a sentence. We use a bag-of-words representation for a document and ignore sentences from which we do not extract any tuples. The LDA model assigns each word to a topic. We count the

occurrences of all topics in one sentence and pick the most frequent one as its theme. This feature captures the intuition that different words can indicate the same sense, for example, “film,” “show,” “series” and “television” are about “entertainment,” while “coach,” “game,” “jets,” “giants” and “season” are about “sports.”

5.3.2 Sense clusters and relation clusters

For the sense disambiguation model, we set the number of topics (senses) to 50. We experimented with other numbers, but this setting yielded the best results based on our automatic evaluation measures. Note that a path has a multinomial distribution over 50 senses but only a few senses have non-zero probabilities.

We look at some sense clusters of paths. For path “A play B”, we examine the top three senses, as shown in Table 5.1. The last two senses “entertainment” and “music” are close. Randomly sampling some entity pairs from each of them, we find that the two sense clusters are precise. Only 1% of pairs from the sense cluster “entertainment” should be assigned to the “music” sense. For the path “play A in B” we discover two senses which take the most probabilities: “sports” and “art.” Both clusters are precise. However, the “sports” sense may still be split into more fine-grained sense clusters. In “sports,” 67% pairs mean “play another team in a location” while 33% mean “play another team in a game.”

We also closely investigate some relation clusters, shown in Table 5.2. Both the first and second relation contain path “A play B” but with different senses. For the second relation, most paths state “play” relations between two teams, while a few of them express relations of teams acquiring players from other teams. For example, the entity pair “(Atlanta Hawks, Dallas Mavericks)” mentioned in sentence “The Atlanta Hawks acquired point guard Anthony Johnson from the Dallas Mavericks.” This is due to that they share many entity pairs of team-team.

5.3.3 Baselines

We compare our approach against several baseline systems, including a generative model approach and variations of our own approach.

Rel-LDA: Generative models have been successfully applied to unsupervised relation extraction [Rink and Harabagiu, 2011, Yao et al., 2011]. We compare against one such model: An extension to standard LDA that falls into the framework presented by Yao et al. [2011]. Each document consists of a list of tuples. Each tuple is represented by features of the entity pair, as listed in §5.2.1, and the path. For each document, we draw a multinomial distribution over relations. For each tuple, we draw a relation topic and independently generate all the features. The intuition is that each document discusses one domain, and has a particular distribution over relations.

In our experiments, we test different numbers of relation topics. As the number goes up, precision increases whereas recall drops. We report results with 300 and 1000 relation topics.

One sense per path (HAC): This system uses only hierarchical clustering to discover relations, skipping sense disambiguation. This is similar to DIRT [Lin and Pantel, 2001]. In DIRT, each path is represented by its entity arguments. DIRT calculates distributional similarities between different paths to find paths which bear the same semantic relation. It does not employ global topic model features extracted from documents and sentences.

Local: This system uses our approach (both sense clustering with topic models and hierarchical clustering), but without global features.

Local+Type This system adds entity type features to the previous system. This allows us to compare performance of using global features against entity type features. To determine entity types, we link named entities to Wikipedia pages using the Wikifier [Ratinov et al., 2011] package and extract categories from the Wikipedia page. Generally Wikipedia provides many types for one entity. For example, “Mozart” is a

person, musician, pianist, composer, and catholic. As we argued in §5.1, it is difficult to determine the right granularity of the entity types to use. In our experiments, we use all of them as features. In hierarchical clustering, for each sense cluster of a path, we pick the most frequent entity type as a feature. This approach can be seen as a proxy to ISP [Pantel et al., 2007], since selectional preferences are one way of distinguishing multiple senses of a path.

Our Approach+Type This system adds Wikipedia entity type features to our approach. The Wikipedia feature is the same as used in the previous system.

5.3.4 Automatic Evaluation against Freebase

We evaluate relation clusters discovered by all approaches against Freebase. We use coreference evaluation metrics: pairwise F-score and B^3 [Bagga and Baldwin, 1998]. Pairwise metrics measure how often two tuples which are clustered in one semantic relation are labeled with the same Freebase label. We evaluate approximately 10,000 tuples which occur in both our data and Freebase. Since our system predicts fine-grained clusters comparing against Freebase relations, the measure of recall is underestimated. The precision measure is more reliable and we employ F-0.5 measure, which places more emphasis on precision.

Matthews correlation coefficient (MCC) [Baldi et al., 2000] is another measure used in machine learning, which takes into account true and false positives and negatives and is generally regarded as a balanced measure which can be used when the classes are of very different sizes. In our case, the true negative number is 100 times larger than the true positive number. Therefore we also employ MCC, calculated as

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

The MCC score is between -1 and 1. The larger the better. In perfect predictions, FP and FN are 0, and the MCC score is 1. A random prediction results in score 0.

Table 5.3 shows the results of all systems. Our approach achieves the best performance in most measures. Without using sense disambiguation, the performance of hierarchical clustering decreases significantly, losing 17% in precision in the pairwise measure, and 15% in terms of B^3 . The generative model approach with 300 topics achieves similar precision to the hierarchical clustering approach. With more topics, the precision increases. However, the recall of the generative model is much lower than those of other approaches. We also show the results of our approach without global document and sentence theme features (Local). In this case, both precision and recall decrease. We compare global features (Our approach) against Wikipedia entity type features (Local+Type). We see that using global features achieves better performance than using entity type based features. The entity type features do not help much, due to that we cannot determine which particular type to choose for an entity pair. Take pair “(Hillary Rodham Clinton, Jonathan Tasini)” as an example, choosing *politician* for both arguments instead of *person* will help.

We should note that these measures provide comparison between different systems although they are not accurate. One reason is the following: some relation instances should have multiple labels but they have only one label in Freebase. For example, instances of a relation that a person “was born in” a country could be labeled as “/people/person/place_of_birth” and as “/people/person/nationality.” This decreases the pairwise precision. Further discussion is in § 5.3.6.

5.3.5 Path Intrusion

We also evaluate coherence of relation clusters produced by different approaches by creating path intrusion tasks Chang et al. [2009]. In each task, some paths from one cluster and an intruding path from another are shown, and the task is to identify one single path which is out of place. For each path, we also show one example sentence. We ask three graduate students in natural language processing to identify

System	Pairwise				B^3		
	Prec.	Rec.	F-0.5	MCC	Prec.	Rec.	F-0.5
Rel-LDA/300	0.593	0.077	0.254	0.191	0.558	0.183	0.396
Rel-LDA/1000	0.638	0.061	0.220	0.177	0.626	0.160	0.396
HAC	0.567	0.152	0.367	0.261	0.523	0.248	0.428
Local	0.625	0.136	0.364	0.264	0.626	0.225	0.462
Local+Type	0.718	0.115	0.350	0.265	0.704	0.201	0.469
Our Approach	0.736	0.156	0.422	0.314	0.677	0.233	0.490
Our Approach+Type	0.682	0.110	0.334	0.250	0.687	0.199	0.460

Table 5.3. Pairwise and B^3 evaluation for various systems. Since our systems predict more fine-grained clusters than Freebase, the recall measure is underestimated.

the intruding paths. We use the majority vote strategy to resolve disagreements.

Table 5.4 shows one example intrusion task.

Path	Example sentence
beat	Dmitry Tursunov beat the best American player, Andy Roddick
who lose to	Sluman , Loren Roberts (who lost a 1994 Open playoff to Ernie Els at Oakmont ...
who beat	... offender seems to be the Russian Mariya Sharapova , who beat Jelena Dokic
<i>a broker at</i>	<i>Robert Bewkes, a broker at UBS for 12 years</i>
meet	Howell will meet Geoff Ogilvy , Harrington will face Davis Love III

Table 5.4. A path intrusion task. We show 5 paths and ask the annotator to identify one path which does not belong to the cluster. And we show one example sentence for each path. The entities (As and Bs) in the sentences are bold. And the italic row here indicates the intruder.

From Table 5.5, we see that our approach achieves the best performance. We concentrate on some intrusion tasks and compare the clusters produced by different systems.

The clusters produced by HAC (without sense disambiguation) are coherent if all the paths in one relation take a particular sense. For example, one task contains paths “A, director at B”, “A, specialist at B”, “A, researcher at B”, “A, B professor” and “A’s program B”. It is easy to identify “A’s program B” as an intruder when the annotators realize that the other four paths state the relation that people work in

System	Correct
Rel-LDA/300	0.737
Rel-LDA/1000	0.821
HAC	0.852
Local+Type	0.773
Our approach	0.887

Table 5.5. Results of intruding tasks of all systems.

an educational institution. The generative model approach produces more coherent clusters when the number of relation topics increases.

The system which employs local and entity type features (Local+Type) produces clusters with low coherence because the system puts high weight on types. For example, (*United States*, A talk with B, *Syria*) and (*Canada*, A defeat B, *United States*) are clustered into one relation since they share the argument types “country”-“country”. Our approach using the global theme features can correct such errors.

5.3.6 Error Analysis

We also closely analyze the pairwise errors that we encounter when comparing against Freebase labels. Some errors arise because one instance can have multiple labels, as we explained in §5.3.4. One example is the following: our approach predicts that (*News Corporation*, buy, *MySpace*) and (*Dow Jones & Company*, the parent of, *The Wall Street Journal*) are in one relation. In Freebase, one is labeled as “/organization/parent/child”, the other is labeled as “/book/newspaper_owner/newspapers_owned”. The latter is a sub-relation of the former. We can overcome this issue by introducing hierarchies in relation labels.

Some errors are caused by selecting the incorrect sense for an entity pair of a path. For instance, we put (*Kenny Smith*, who grew up in, *Queens*) and (*Phil Jackson*, return to, *Los Angeles Lakers*) into the “/people/person/place_of_birth” relation

cluster since we do not detect the “sports” sense for the entity pair “(Phil Jackson, Los Angeles Lakers).”

5.4 Related work

There has been considerable interest in unsupervised relation discovery.

Our work is closely related to DIRT [Lin and Pantel, 2001]. Both DIRT and our approach represent dependency paths using their arguments. Both use distributional similarity to find patterns representing similar semantic relations. Based on DIRT, Pantel et al. [2007] addresses the issue of multiple senses per path by automatically learning admissible argument types where two paths are similar. They cluster arguments to fine-grained entity types and rank the associations of a relation with these entity types to discover selectional preferences. Selectional preferences discovery Ritter et al. [2010], Seaghdha [2010] can help path sense disambiguation, however, we show that using global features performs better than entity type features.

Our approach is also related to feature partitioning in cross-cutting model of lexical semantics Reisinger and Mooney [2011]. Our sense disambiguation model is inspired by this work. There they partition features of words into views and cluster words inside each view. In our case, each sense of a path can be seen as one view. However, we allow different views to be merged since some views overlap with each other.

Clustering approaches are explored in relation extraction [Hasegawa et al., 2004, Hachey, 2009, Bollegala et al., 2010], however these approaches neither deal with polysemy nor incorporate global features, such as sentence and document themes.

Many generative probabilistic models have been applied to relation extraction. For example, varieties of topic models are employed for both open domain [Yao et al., 2011] and in-domain relation discovery [Chen et al., 2011, Rink and Harabagiu, 2011]. Our approach employs generative models for path sense disambiguation, which

achieves better performance than directly applying generative models to unsupervised relation discovery.

5.5 Conclusion

We explore senses of paths to discover semantic relations. We employ a topic model to partition entity pairs of a path into different sense clusters and use hierarchical agglomerative clustering to merge senses into semantic relations. Experimental results show our approach discovers precise relation clusters, and outperforms a generative model approach and a clustering method which does not address sense disambiguation. We also show that using global features improves the performance of unsupervised relation discovery over using entity type based features.

CHAPTER 6

UNIVERSAL SCHEMA FOR ENTITY TYPE CLASSIFICATION

We now begin to describe our core work with universal schema. We describe the simple case, universal schema for entity types in this chapter. Next chapter we describe universal schema for relation extraction.

Learning entity types is useful in many applications, such as knowledge base construction, relation extraction, and query intent prediction. Fine-grained entity type ontologies are especially valuable, but typically difficult to design because of endless quandaries about level of detail and boundary cases. In this chapter, we present *universal schema* for automated fine-grained entity type prediction. The set of types is taken as the union of textual surface patterns (e.g. appositives) and pre-defined types from available databases (e.g. Freebase)—yielding not tens or hundreds of types, but more than ten thousand entity types, such as financier, criminologist, and musical trio. We robustly learn mutual implications among this large union by probabilistic matrix factorization, thus avoiding the need for hand-labeled data. Experimental results demonstrate significant improvement over classification based approaches on predicting fine-grained entity types. Experiments also show that our predicted entity types can benefit downstream applications, such as relation extraction [Yao et al., 2013].

6.1 Introduction

Knowledge about the underlying things in the world (such as people, places, plants, and products) rather than merely character strings (like pages or paragraphs)

enables deeper, more structured understanding of the world. The significant resources being devoted to Google’s *Knowledge Graph*, Facebook’s *Graph Search*, and Microsoft’s *Satori* are testaments to the importance of modeling the world as entities and the relations among them. One of the first, fundamental tasks when dealing with entities is to predict their categories or “types.”

Entity types can be useful in many applications. In some cases, such as relation extraction [Yao et al., 2010, Roth and Yih, 2007] or query intent discovery [Cheung and Li, 2012, Pantel et al., 2012, Balog and Neumayer, 2012], entity types are hidden variables included to improve accuracy on the target task. In other cases, such as knowledge base construction, entity types may be a prominent user-visible feature [Carlson et al., 2010, Hoffart et al., 2012], where they help users browse or find entities more easily, and visualize them better.

Occasionally the ontology of entity types is coarse, such as the four types in the CoNLL-2003 shared task (person, organization, location and miscellaneous), but often finer-grained ontologies are more useful. For example, specializations of people, including politician, scientist, and athlete are defined in previous work [Fleischman and Hovy, 2002, Giuliano and Gliozzo, 2008, Ekbal et al., 2010]. Others are even more detailed; for instance, the Unified Medical Language System (UMLS) defines an ontology of 987,321 biomedical concepts. Defining such ontologies is a significant challenge, often giving rise to debates about desired granularity and subtle questions about boundary cases. These difficulties appear both when the assignment of entities to types is exclusive and when it is one-to-many.

Once the ontology is defined, the problem of building the automated classification system remains. The most common approach is supervised training from a set of entity mentions labeled into the ontology [Fleischman and Hovy, 2002, Tanev and Magnini, 2006]. However labeling such data is painful—especially with fine-grained ontologies. Furthermore, when the ontology evolves or expands (as it often does),

the data labeling must be re-visited. Even when used as hidden variables, the set of entity types may warrant adjustment because an ontology tuned to the task at hand typically performs better—for example, Pantel et al. [2007] show that the entity types in the WordNet ontology [Fellbaum, 1998] are not as effective as those derived from automatic clustering for the task of learning selectional preferences. Unsupervised clustering may be employed to derive entity types [Elsner et al., 2009, Pantel et al., 2007, Yao et al., 2011], but the resulting types often have peculiar, undesirable, only weakly interpretable boundary and granularity choices.

This chapter presents an approach to fine-grained entity type prediction that avoids the need to manually design an ontology, avoids the need for labeled data, and avoids the boundary difficulties that arise from forcing our semantics into finite, pre-defined, somewhat arbitrary “boxes.” Our approach is *universal schema*, that defines types as the *union* of all available types from all input sources, including multiple pre-existing ontologies and naturally-occurring textual surface-form expressions that indicate entity type, such as appositives, isa-expressions, or even adjectival or verb phrases. For example, “James Cameron” may appear as a *person/director* in Freebase [Bollacker et al., 2008], and as a *person* in TAC/KBP.¹ He may also occur in text documents as a *movie-mogul*, *Canadian citizen*, and *jerk* in clauses like “James Cameron, a movie-mogul,” “James Cameron, a Canadian citizen” and “James Cameron is a jerk.” Rather than five, fifty, or five-hundred entity types, this universal schema approach typically yields more than ten thousand entity types, particularly from textual surface forms. Universal schema does not force the natural diversity and ambiguity of the original input types into a smaller set of types. Universal schema gets to the heart of the taste by predicting sources instead of latent variables that we do not know the ground truth.

¹<http://www.nist.gov/tac/2013/KBP/>

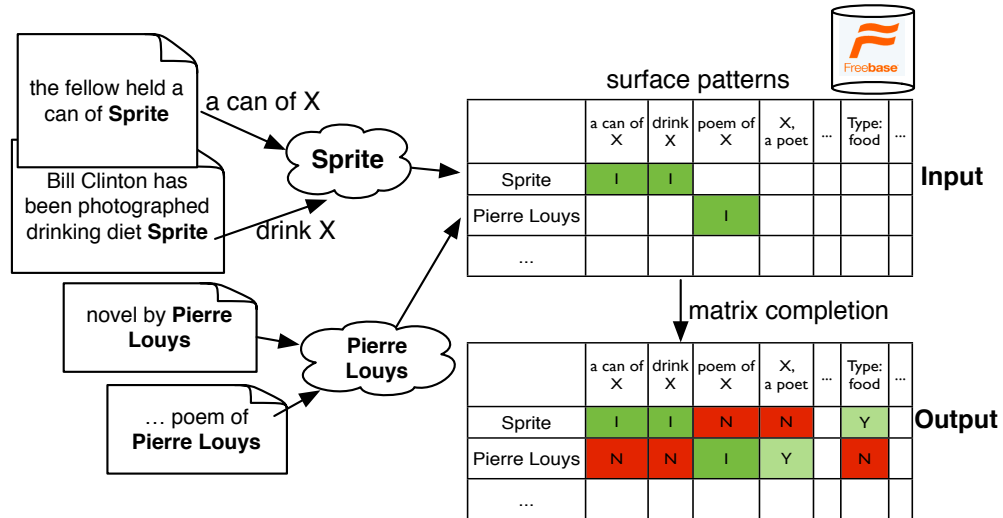


Figure 6.1. Overview of our system.

The key characteristic of universal schema is that it enables us to model directed implicature among the many candidate types of an entity. We cast the problem as a large matrix completion task. Each row in the matrix corresponds to an entity, and each column an entity type. Some cells of the matrix are observed and marked true, and many are unobserved. It is the job of matrix completion to “fill in” the matrix, marking the unobserved cells as either true or false. For example, although we may not have directly observed that “Barack Obama” is a *leader*, our model will infer it by having observed that he is a *president* and *commander-in-chief*—doing so by leveraging various patterns of co-occurrences among these types in other entities. Similarly it will infer that he is neither a *movie-mogul* nor a *waterfall*.

We achieve this using probabilistic matrix factorization—efficiently estimating vector embeddings for both entities and types by online stochastic gradient descent optimization. The probability of assigning a type to an entity is determined by the dot-product of the corresponding embeddings, mapped through a logistic function. The output matrix can be queried as a probabilistic database since each cell has a probability score.

Figure 6.1 shows the overview of our system. Our information sources include text documents and existing knowledge bases, such as Freebase. Note that documents can be from any sources, including the web, a newswire corpus, and so on. From the text documents we identify entity mentions and extract the dependency path as the surface patterns for them. To fill in our matrix, we perform string match coreference to cluster mentions into entities. In figure 6.1, we show that two mentions of “Sprite” that refer to the same entity are clustered to one row in the matrix. We also link these entities to knowledge bases. We produce the output matrix using probabilistic matrix factorization—efficiently estimating vector embeddings for both entities and types by online stochastic gradient descent optimization.

Table 6.1 shows some examples from our data: the first column lists several entities, the second column shows observed types for the corresponding entity and the third column shows newly predicted types. In one example, we observe “Sprite, a drink,” “(subject) drink Sprite” and so on in text, we predict “Sprite” is *food.beverage* in Freebase. In another example, seeing “poem of Pierre Louys” and “novel by Pierre Louys,” we predict “Pierre Louys, a poet,” “Pierre Louys, a novelist,” and “Pierre Louys, an author.”

We carry out experiments on various datasets. On a small closed dataset [Ling and Weld, 2012], we show that our approach is a general framework that can achieve comparable performance as the baseline classifier. On a large scale web data with entities linked to Wikipedia [Singh et al., 2012], we show that our approach is scalable and can achieve better performance than baseline models. Using New York Times data [Sandhaus, 2008], we show that entity type predictions can help relation extraction. In all of the experiments, we employ Freebase as the knowledge base for entity and relation types.

Entity	Observed	Predicted
Mohamed al-Fayed	tycoon, owner, entrepreneur, financier estate of X, purchase by X	billionaire, magnate, person:Freebase
House of Pain	X’s member, reminiscent of X, music by X, rap group X	trio, band, X rap singer of X, rapper X
Sprite	commercial, drink, brand, ad drink X, campaign for X	beverage:Freebase
Jonathan Wolken	founder, director, choreographer X’s solo, create by, choreograph by	dancer dance by X
Pierre Louys	poem of, novel by, ’s poem, ’s novel	poet, novelist, author
Rick Wamsley	goaltender, goalie	goalkeeper
Walter Arndt	scholar, libretto by, translate by	translator

Table 6.1. Some entities, observed types and predicted ones by our system. We can describe an entity in any granularity based on the patterns or types from ontologies. The patterns are translated from dependency parsing paths as described in §6.3.

6.2 Factorization Models

In this section, we describe matrix factorization models for entities and types. Note that types can be understood as unary relations, so we use unary relations and types interchangeably. Our observations are entities from text. Each cell represents an entity and a unary relation that holds according to our data source. Our goal is to predict new unary relations that also hold for the entity.

Our technical approach is based on extensions to probabilistic models of matrix factorization and collaborative filtering [Collins et al., 2001, Rendle et al., 2009]. In collaborative filtering, items are recommended to users based on collecting many users’ ratings about the items. For example, if both user X and Y like item A, and user X likes item B, it is likely that user Y likes item B as well. In our scenario, an entity corresponds to a user; a unary relation corresponds to an item; and an observed cell corresponds to a positive rating by the user for the item. By collecting information about preferences of other entities, we can “recommend” unary relations that hold for an entity. Our approach is novel in that we learn implication among these unary relations.

We fill a matrix $|E| \times |R|$ with unary relation instances, where E corresponds to entities and R to unary relations. Assume we index an entity with e and a relation with r . Each matrix cell is a binary variable, denoted as $x_{e,r}$. The variable is 1 when relation r holds for entity e , and 0 otherwise. For example, observing “Rick Wamsley, a goaltender,” we fill the corresponding cell (Rick Wamsley, “X, a goaltender”) with 1. Note that all our training cells are positive, as one rarely states explicitly that a particular unary relation does not hold.

In our matrix factorization approach, we embed each entity and relation as latent vectors \mathbf{a}_e and \mathbf{v}_r in a K -dimensional space, respectively. Each dimension is a component (c). Since square loss is not appropriate for discrete data, a logistic regression version of matrix factorization is a better choice for our binary data [Collins et al., 2001]. Thus we have:

$$\begin{aligned}\theta_{e,r} &= \sum_c a_{e,c} v_{r,c} \\ x_{e,r} &= \sigma\left(\sum_c a_{e,c} v_{r,c}\right)\end{aligned}$$

The first formula is factorizing a matrix into a multiplication of two matrices. In matrix representation, it is $\Theta = AV$. The second is applying a logistic function $\sigma(\theta) = 1/(1 + \exp(-\theta))$ to $\theta_{e,r}$ to model a binary cell. This has a probabilistic interpretation: each cell is drawn from a Bernoulli distribution with natural parameter θ .

To learn low dimensional representations, we maximize the log likelihood of the observed cells under the probabilistic model above. Notice that in our training data we only observe positive cells and have no accurate data on which relations do *not* hold for an entity. However, learning requires negative training data. We address this issue by sampling unobserved relations for an entity based on their frequencies in the whole dataset and treating them as negative. The joint probability of all cells

is defined as:

$$\begin{aligned} & \prod_{cell} p(x_{cell} = 1)^{\delta(x=1)} (1 - p(x = 0))^{\delta(x=0)} \\ &= \prod_{cell} \sigma(\theta)^{\delta(x=1)} (1 - \sigma(\theta))^{\delta(x=0)} \end{aligned}$$

For simplicity, we elide the subscript of each cell, and $\delta(x = 1)$ stands for the number of positive cells.

To simplify the joint probability, we can represent negative cells as positive cells by choosing a different natural parameter. Thus the joint probability becomes $\prod_{cell} \sigma(\theta)$. For simplicity, we use Θ to represent all the parameters. Adding a prior for the parameters, we can write the log likelihood as

$$\log \sum_{cell} \sigma(\theta) - \Lambda_{\Theta} \|\Theta\|^2.$$

The gradient with respect to θ is:

$$(1 - \sigma(\theta)) \nabla \theta - \lambda \theta$$

Taking gradients of θ with respect to the parameters, we obtain:

$$\nabla a_{e,c} = v_{r,c}$$

$$\nabla v_{r,c} = a_{e,c}$$

We notice that, in our data, some unary relations are more popular than others. In order to capture this, we introduce a bias for each relation (b_r). We also introduce a bias shared across the whole matrix (b). Finally we have

$$\theta_{e,r} = \sum_c a_{e,c} v_{r,c} + b_r + b$$

We use stochastic gradient optimization to effectively deal with the large scale of our matrices. In each iteration, we traverse random permutations of all training cells, randomly sample some negative cells for each training cell, and update the corresponding \mathbf{a}_e and \mathbf{v}_r vectors for the positive and negative cells based on their corresponding gradients.

We update the parameters of a positive cell (e, r) using the following formulas, iterating over each component c , with learning rate l :

$$a_{e,c} = a_{e,c} + l((1 - \sigma(\theta))v_{r,c} - \lambda a_{e,c}) \quad (6.1)$$

$$v_{r,c} = v_{r,c} + l((1 - \sigma(\theta))a_{e,c} - \lambda v_{r,c}) \quad (6.2)$$

Likewise for a negative cell, we update the parameters using:

$$a_{e,c} = a_{e,c} + l((0 - \sigma(\theta))v_{r,c} - \lambda a_{e,c}) \quad (6.3)$$

$$v_{r,c} = v_{r,c} + l((0 - \sigma(\theta))a_{e,c} - \lambda v_{r,c}) \quad (6.4)$$

To predict a cell, we calculate $x_{e,r} = \sigma(\theta_{e,r})$. In our experiments, cells with scores above a threshold are considered as true.

6.2.1 Neighbor Model

In our matrix, each column represents one entity type. There is also other information that can be useful for entity type prediction. For example, the surface string tokens, the head words, the context words of an entity [Ling and Weld, 2012]. To incorporate these information resources into our model, we employ the Neighbor Model, a model that is analogous to classifiers [Koren, 2008]. Here we represent the

information sources as features and other columns representing semantic types of entities as labels. The score of each cell is defined as:

$$\theta_{e,r} = \sum_i w_i f_i(e, r)$$

In the equation, $f_i(e, r)$ defines a conjunctive feature, i.e. $token = Department, label = university$. During training, we learn the weight w_i for each feature.

In our experiments, we also introduce a combined factorization and neighbor model, and each cell has its θ defined as:

$$\theta_{e,r} = \sum_c a_{e,c} v_{r,c} + b_r + b + \sum_i w_i f_i(e, r)$$

It is straightforward to calculate the gradients for weights of features. Therefore we still use maximum log likelihood as the objective function and employ stochastic gradient descent to learn the parameters.

6.3 Experiments

Our goal is to predict types for entities, i.e. missing cells in the entity-relation matrix. In the following, we design experiments to measure the accuracy of these predictions. We analyze the embeddings of patterns learned by our model, evaluate the predicted entity types, and also experiment with using our predicted entity types for relation extraction.

6.3.1 Data Sets

We perform experiments on various data sets, including NYT data, a small closed data set, and a large scale web data set. As a knowledge base, we use Freebase for entity and relation types in the experiments.

New York Times and Freebase data. We extract unary relations from New York Times data for the years 1990 to 2007 [Sandhaus, 2008]. We preprocess the documents as described in §2.7. Following Szpektor and Dagan [2008], we extract dependency paths originating from a (named) entity mention as unary relations. Specifically, we traverse from the head token of the entity mention to the root of the dependency tree. Whenever we come across a content word (nouns, adjectives etc.), the current (lexicalized) path from the entity mention to this content word node is used as one unary relation. We stop when approaching a verb or a clause boundary. Additionally, when a verb is encountered, other direct children of the verb are also included in the path. For example, we can have “X buy share,” “X roll over.” This yields many simple relations that could serve as entity types, including appositive structures. For example, the unary relation “X, a magnate” can define “magnate” as the corresponding entity type.

The universal schema approach labels types of entities, not entity mentions. This is inherent in the method, since it learns the embedding that leverages co-occurrences among multiple entity type patterns (coming from multiple mentions) of the same entity. Thus universal schema relies on entity resolution as a prerequisite. However, entity resolution is not our main focus in this paper; and there are many complex alternative methods from which to choose. To ease reproducibility and demonstrate the robustness of our entity type prediction method in this paper we use instead simple string matching to link entity mentions in NYT documents to Freebase entities. Specifically, if the Freebase entity has the same string form as the entity mention, we link them as one entity. If multiple Freebase entities have the same string form, we choose the one that is most popular. For example, “Canada” could be a *country*, or a kind of *wine*. We choose *country*. This method is also employed by previous work [Mintz et al., 2009, Lin and Pantel, 2001, Yao et al., 2010]. We allow one entity to have multiple types, not disambiguating entities that have the same string form.

Integrating universal schema with more complex entity resolution systems is a topic for future work.

On this dataset, we analyze the embeddings of patterns obtained by our factorization model in § 6.3.3. We also test downstream application of predicted entity types, i.e., taking them as input for relation extraction (see § 7.3.6).

Local reports data from UW. A practical use case of entity type prediction is that we train a model on available data sources and apply it to new documents that we have not seen before. Specifically, we want to predict types for entities occurring in a closed set of documents. Exhaustive annotation for all entities with all possible types should be available when measuring precision, recall, and F1. We use a recent benchmark that satisfies these conditions, local reports data [Ling and Weld, 2012]. This dataset consists of 18 documents and approximately 430 sentences, and each entity mention is labeled with all possible entity types.

WikiLinks Data. To show that our approach is scalable, we also experiment with a large scale web data set. This dataset, originally used for large scale coreference [Singh et al., 2012], contains a large number of entity mentions collected from web pages. The advantage of using this dataset is that we have labeled coreference for these entity mentions since the creators of this dataset use Wikipedia links to annotate whether two mentions are referring to the same entity. Sentences to the left and right of each mention are available as contexts. Dependency parse trees are not provided. We extract the left and right words to the current mention as unary relations. We heuristically filter out some entities, such as entities that contains digits, and entities that are not proper nouns. We also filter out infrequent unary relations.

As a result, we obtain 847,039 entities, 23,810 unary relations, and 9,427,543 observed cells in the input matrix. Since part of Freebase data is from Wikipedia, we use mappings from Wikipedia links to Freebase entities to include entity types from the ontology. Approximately 55% of entities in this dataset have Freebase labels. In total,

we have 513 entity types from Freebase. These types are moderately fine-grained, for example, *wine.grape*, *food.cheese*, *sports.boxer*, and *event.disaster*. Predicting these types are non-trivial since many of them do not have many training instances.

6.3.2 Baselines

We compare our approach against binary classifiers (Classifier) as traditionally used in distant supervision [Mintz et al., 2009], leveraging a knowledge base as a supervision source and employing a one-vs-all classification strategy, considering one type as positive, all the others as negative. As the classifier, we use a log-linear model trained by maximum log likelihood and L2 regularization.

On local reports data, we also compare against a multi-instance multi-label classifier for entity prediction (UW) [Ling and Weld, 2012]. This approach performs perceptron style training, but uses a set of positive labels instead of one positive label. Each update discourages the model from predicting labels that are not in the positive set, and increases weights for labels in the positive set.

We have several variations for universal schema. We employ the factorization model (F), the neighbor model (N) and the combined model (F+N).

In the following, we will describe our experiments on these data sets.

6.3.3 Pattern Analysis on NYT data

We begin by providing some intuition for the embeddings learned for NYT data. On this dataset, we obtain 503,301 entities and 16,916 patterns. Our model learns a low dimensional vector for each pattern. It is usually challenging for humans to interpret these vectors. Intuitively, patterns representing similar entity types should be close to each other in the low dimensional space. To demonstrate that this happens, we perform hierarchical agglomerative clustering using cosine similarity as distance measure on these vectors. We query some patterns and show in Table 6.2 the clusters in which they occur. We observe here that our approach can learn diverse and accurate

Target	Patterns
magnate	developer, investor, estate, financier, owner, partner, shareholder, tycoon, landlord, billionaire, buyer, principal, capitalist
band	tune, album, country, trio, duo, blues, rock, folk, singer of, wing with, music of, tour with, musician like, hit for sound of, recording by, song by, act like, 's singer, 's song
actor	performer, co-star, X appear in, X portray in, X is cast as, X play character, X play in, star X as, feature X as/in, X reprise embody by, film with, star with, actor like, act by, narrate by, play by
player	draft, guard, defensive, lineman, linebacker, fullback, quarterback X miss after, X recover, X tear in, X injure knee, X is suspended for X pick off, X's interception, X run yard, X has catch, X's touchdown, tackle X, recover by X, return by X

Table 6.2. Top similar patterns to the target queries.

patterns that are indicative of the target patterns. For example, we learn different roles of players such as “lineman” and “quarterback.” We also find actions of players, such as “injure knee,” “has catch,” and “run yard.”

We also compare against features from traditionally learned one-vs-all classifiers on the same dataset, using query patterns as labels. We list top ranked features of each binary classifier for a target pattern in Table 6.3. Note that the classifiers do not find as many high quality patterns as our approach. The classifiers instead often find frequent patterns that co-occur with the target pattern.

6.3.4 Closed Set Evaluation

We compare our method against both the Classifier and UW on the closed local reports dataset. The UW approach employs a multi-class multi-label classifier for fine-grained entity recognition. Their model labels entity mentions, not entities. We obtain two results from this baseline, one that uses string match for coreference to translate their labels on mentions into multiple labels on entities, for purposes of comparison (UW-mention); the other performs coreference using string match up front and applies their approach on the resulting entities instead of mentions (UW).

Target	Top features
magnate	magnate X, businessman, tycoon, developer, chairman, who was born, X is a/the buyer, X purchase, control by X, name for X, form by X, house of X, son of X, widow of X
band	band X, singer X, record, outfit, open for X, player for X, X open show, X is taught, X featuring, X share, X mix, X turn for, X sell copy, X play song, X's producer, X entertain,
actor	actor X, television, X is an actor, X is outstanding, X is eager, cast as X, play by X, marry to X, star as X , ask X on
player	bassist, athlete, foot, X which is percent, X's note

Table 6.3. Top ranked patterns learned by the baseline classifiers. Not all patterns can imply the target patterns. The patterns are not as diverse as patterns learned by our approach.

In addition to dependency path, features including words and head words of the entity, as well as the contextual unigrams and bigrams, are also beneficial for predicting entity types. These features are used in UW [Ling and Weld, 2012]. We incorporate them using our neighbor model (see Section 6.2.1).

Our approach (Universal), the distant supervision classifier baseline (Classifier), and UW [Ling and Weld, 2012] use the same training data, linking Freebase entities to mentions from Wikipedia articles. The input matrix in our approach has approximately 623K entities.

Table 6.4 shows the F1 scores of different systems. Our approach is consistently better than the distant supervision classifier, better than applying UW system on entities instead of mentions, and comparable with the UW approach when translating their predictions on mentions to predictions of entities. We report the results of combined model for universal schema. Only using the factorization model is unfair on this dataset, since the factorization model does not consider the contextual features. However, this evaluation is somewhat deficient because it is based on a relatively small data set. Hence we are interested in applying our approach to WikiLinks data.

System	Precision	Recall	F1
Classifier	0.585	0.438	0.501
UW	0.582	0.475	0.523
UW-mention	0.521	0.590	0.553
Universal (F+N)	0.633	0.492	0.553

Table 6.4. Performance on predicting Freebase entity types on a closed data set.

System	Precision	Recall	F1
Classifier	0.619	0.215	0.320
N	0.246	0.450	0.318
F	0.215	0.355	0.268
F+N	0.303	0.427	0.354

Table 6.5. Performance on predicting Freebase entity types on WikiLink.

6.3.5 Evaluation on WikiLinks

On WikiLink data we perform held-out evaluation, using 80% of the entities as training data, and the remaining 20% as test data. Freebase labels in the test set are hidden and to be predicted for evaluation.

In the neighbor model, we consider all the columns of patterns in one row as features, and types from Freebase as labels. Likewise for the combined model. We list the results in Table 6.5. We can see that Classifier has high precision and all variations of our approach have higher recall than Classifier. Our neighbor model performs similarly to Classifier in F1. Our combined model outperforms both. Note that Freebase is not complete. There exist some entities that our systems predict correctly but are not annotated in Freebase. When this occurs, it leads to underestimation of precision.

Looking at the 513 entity types we have from Freebase we find that some of them have an exactly corresponding word pattern type in a column of our matrix. This alignment provides yet another avenue for evaluation. For example we can evaluate

Query	N	F	F+N
wine.grape	0.538	0.403	0.523
sports.boxer	0.650	0.677	0.721
food.cheese	0.228	0.095	0.187
physician	0.500	0.539	0.522
food.beverage	0.294	0.205	0.324
disaster	0.437	0.362	0.465
drug	0.112	0.175	0.209
software	0.423	0.500	0.500

Table 6.6. F1 measure on some human annotated fine-grained types. We take these types as representatives of pattern based types.

whether the entities labeled with the word pattern “boxer” are labeled with the *sports.boxer* type in Freebase. We obtain the set of “boxer” candidate entities by taking the union of predicted “boxers” from all evaluated systems. Labeled truth is determined by WikiLinks’ linkage to Freebase and human annotations based on Wikipedia articles. We perform this evaluation for eight randomly-selected prominent types, gathering different sets of entities for each query, and within each checking the prediction of only one type that defines the query. Table 6.6 shows the results. As in the previous experiment we find that F+N performs best.

We also perform error analysis to search for commonalities among our error cases. We find that one common case of error occurs when context words are insufficient. For example, criminals who killed people may be assigned the type *disaster* because “kill” occurs as a context word; or patients treated with medicine or medical tests sometimes may be assigned the type *drug* since “vaccine” occurs as a context word. Dependency parse information can help in these cases; “treat X with vaccine” can indicate that X is not a *drug*. However, it is challenging to obtain accurate dependency parse trees on some data, like sentences from web pages. Entity type exclusive constraints can help as well, knowing that X is a *person* and a *person* cannot be a *drug* can correct these predictions. Other error cases are simply caused by co-occurring confounding

features; for example researchers in chemistry are sometimes predicted to have type *physician*.

6.3.6 Parameter Selection

For all experiments, we vary the number of components in $\{50, 100, 150, 200\}$, learning rate in $\{0.02, 0.05, 0.1\}$, regularizers for embedding vectors of entities and relations in $\{0.01, 0.02, 0.05\}$, number of negative examples in $\{1,3,5,7,10\}$. We use stochastic gradient descent as our optimization method. This method efficiently learns from examples one at a time, therefore scaling well to large datasets.

6.4 Related Work

Classifying entities into large ontologies is a common task and is widely acknowledged as useful. Some researchers have explored entity type classification specifically for categories of people [Fleischman and Hovy, 2002, Giuliano and Gliozzo, 2008, Ekbal et al., 2010]. Large-scale knowledge bases, such as Freebase and its fine-grained entity types, have significant collections of entities that can be used for training traditional classification methods by distant supervision [Ling and Weld, 2012]. Others have also performed entity type classification with a multi-label classifier in a hierarchy of types [Yosef et al., 2012].

The main differences to our approach are: (1) we use matrix factorization rather than classification as the framework for our model, and more importantly; (2) we do not restrict ourselves to predefined entity types, instead leveraging the wide diversity of naturally available data. Even when a pre-existing knowledge base can provide supervision for a classifier, the resulting entity type classifier is still limited by the types envisioned by the creators of the knowledge base ontology. Furthermore, note that even when the goal is merely classification into a specific ontology, matrix fac-

torization’s striving to predict many other text-based entity types provides a kind of multi-task learning [Caruana, 1993] that can be beneficial.

Our work is also related to semantic inference over text [Dagan et al., 2005]. Szpektor and Dagan [2008] aim to discover implications among unary patterns for predicting new unary facts. We have a similar goal here. They concentrate on verb-triggered patterns, whereas we focus on patterns that define entity types, including noun-triggered patterns (such as appositives) and verb-triggered patterns. They employ distributional similarity where we use matrix factorization.

6.5 Conclusion

This chapter presents *universal schema* for assigning entities into multiple entity types on NYT data. We use the term “universal” because the set of types is formed by the union of textual surface patterns and multiple input entity type ontologies. We evaluate our approach on predicting types from ontologies since we have ground truth. On a larger data set, we achieve better performance against an advanced multi-instance multi label classifier; on a small closed dataset, we perform better than a maximum entropy classifier and comparable with the advance classifier.

CHAPTER 7

UNIVERSAL SCHEMA FOR RELATION EXTRACTION

We introduced universal schema for entity types in Chapter 6. We explore universal schema for relation extraction in this chapter.

7.1 Introduction

In previous chapters, we introduce distant supervision for relation extraction. This approach uses a pre-defined, finite and fixed schema of relation types (such as *born-in* or *employed-by*) and relies on the availability of a large database that has the desired schema.

The need for pre-existing datasets can be avoided by using language itself as the source of the schema. This is the approach taken by OpenIE [Banko et al., 2007]. Here surface patterns between mentions of concepts serve as relations. This approach requires no supervision and has tremendous flexibility, but lacks the ability to generalize. For example, OpenIE may find *historian-at*(Ferguson, Harvard) but does not know *is-a-professor-at*(Ferguson, Harvard). OpenIE has traditionally relied on a large diversity of textual expressions to provide good coverage. However this diversity is not always available, and the lack of generalization greatly inhibits its ability to support reasoning.

One way to gain generalization is to cluster textual surface forms that have similar meaning. We described this approach in Chapter 4 and 5. While the clusters discovered by all these methods usually contain semantically related items, closer inspection invariably shows that they do not provide reliable implicature. For exam-

ple, a typical representative cluster may include *historian-at*, *professor-at*, *student-at*, *graduated-from*. Although these relation types are indeed semantically related, note that *professor-at* does not necessarily imply *historian-at*, and *professor-at* certainly does not imply *student-at*. In fact, we contend that any relational schema would inherently be brittle and ill-defined, having ambiguities, problematic boundary cases, and incompleteness. For example, Freebase, in spite of its extensive effort towards high coverage, has no *criticized* nor *student-at* relation.

In response to this problem, we present universal schema. Here we embrace the diversity and ambiguity of original inputs and avoid forcing textual meaning into pre-defined boxes. This is accomplished by defining our schema to be the union of all source schemas: original input forms, e.g. variants of surface patterns similarly to OpenIE, as well as relations in the schemas of many available pre-existing structured databases. Unlike OpenIE, we concentrate on learning asymmetric implicature among relations. This allows us to probabilistically “fill in” inferred unobserved entity-entity relations in this union. For example, after observing *historian-at*(Ferguson,Harvard) our system infers that *professor-at*(Ferguson, Harvard), but not vice versa.

Similar to representing entity instances in a matrix, we represent relation instances as a matrix as well. Here one row stands for one entity tuple instead of one single entity. Each column stands for a binary relation, as opposed to a unary relation in a entity-relation matrix. The rows come from running cross-document entity resolution across pre-existing knowledge bases and textual corpora. The columns come from the union of surface forms and knowledge base relations. We also use a matrix factorization model to learn lower dimensional manifolds for tuples and relations, and a neighbor model to capture local correlations between patterns and knowledge base relations. We still make the binary random variable assumption as in Chapter 6.

We carry out experiments on New York Times articles and Freebase relation instances. We show that our models can accurately predict relationships defined by surface patterns which do not appear explicitly in text, and that learning latent representations of tuples and relations substantially improves results over a traditional classifier approach. Moreover, on predicting relations in Freebase, our model outperforms the current state-of-the-art distant supervision method [Surdeanu et al., 2012] by 10% points Mean Average Precision through joint implication learned among surface patterns and Freebase relations.

7.2 Models

Our observations are relation instances from text and structured data. Each instance is represented by an entity tuple and a relation that holds according to our data source. Our goal is to predict new relations that also hold for the entity tuple.

In this section, we introduce several models that address the task.

7.2.1 Matrix Factorization

Researchers have successfully employed matrix factorization for collaborative filtering. We adapt this model to relation extraction. We organize our observations into a entity-tuple/relation matrix, similar to the entity-relation matrix in Chapter 6. Each row represents an entity tuple and each column represents a relation. The corresponding cell is a binary value, indicating whether the relation holds for the entity tuple. Using t as the tuple/row index, r as the relation/column index, and c as the component index, we have

$$\begin{aligned}\theta_{t,r} &= \sum_c a_{t,c} v_{c,r} \\ \sigma(\theta) &= \frac{1}{1 + \exp(-\theta)}\end{aligned}$$

Similar to matrix factorization for entity instances, taking gradients of θ with respect to the parameters $a_{t,c}$ and $v_{c,r}$, we have:

$$\begin{aligned}\frac{\partial}{\partial a_{t,c}}\theta_{t,r} &= v_{c,r} \\ \frac{\partial}{\partial v_{c,r}}\theta_{t,r} &= a_{t,c}\end{aligned}$$

In our data, some relations are more popular than others. We introduce a bias for each relation (b_r). We also introduce a bias shared across the whole matrix (b). Finally we have

$$\theta_{t,r} = \sum_c a_{t,c}v_{r,c} + b_r + b$$

Similar to matrix factorization for entity instances, we use stochastic gradient optimization to effectively deal with the large scale of our matrices. We update the parameters of a positive cell (t, r) using the following formulas:

$$\begin{aligned}a_{t,c} &= a_{t,c} + l((1 - \sigma(\theta))v_{r,c} - \lambda a_{t,c}) \\ v_{r,c} &= v_{r,c} + l((1 - \sigma(\theta))a_{t,c} - \lambda v_{r,c})\end{aligned}$$

Likewise for a negative cell, we update the parameters using:

$$\begin{aligned}a_{t,c} &= a_{t,c} + l((0 - \sigma(\theta))v_{r,c} - \lambda a_{t,c}) \\ v_{r,c} &= v_{r,c} + l((0 - \sigma(\theta))a_{t,c} - \lambda v_{r,c})\end{aligned}$$

These update formulas are exactly the same as [6.1](#) and [6.3](#), by replacing index e with t . To predict a cell, we calculate $x_{t,r} = \sigma(\theta_{t,r})$. In our experiments, cells with scores above a threshold are considered as true.

7.2.2 Neighbor Model

Matrix factorization captures the global structure of the data, and relations only interact with each other via their low dimensional embeddings. However, the factorization approach fails to capture the local structure of the data. For example, since pattern *champion from* occurs fewer times in the data, the factorization model cannot predict *nationality* correctly. In our data, we observe that even though *champion from* occurs fewer times, it co-occurs with *nationality* frequently. This suggests that in some cases, the truth of one relation only depends on a few other co-occurring relations (neighbors). To capture this localized correlation in our data, we employ neighbor Model (N), a model that is analogous to classifiers [Koren, 2008]. Here we consider current relation as labels, other relations in the same row as features. The score of each cell is defined as:

$$\theta_{t,r} = \sum_i w_i f_i(r', r)$$

In the equation, $f_i(t, r)$ defines a conjunctive feature, i.e. $r' = \text{champion from}$, $r = \text{nationality}$. During training, we learn the weight w_i for each conjunctive feature.

In our experiments, we also introduce a combined factorization and neighbor model, and each cell has its θ defined as:

$$\theta_{t,r} = \sum_c a_{t,c} v_{r,c} + b_r + b + \sum_i w_i f_i(r', r)$$

It is straightforward to calculate the gradients for weights of features. Therefore we still use maximum log likelihood as the objective function and employ stochastic gradient descent to learn the parameters.

7.2.3 Entity Model

Relations have selectional preferences: they allow only certain types in their argument slots. While knowledge bases such as Freebase or DBpedia have extensive

ontologies of types of entities, these are often not sufficiently fine to allow relations to discriminate [Yao et al., 2012b]. Hence, instead of using a predetermined set of entity types, in our entity model, we learn a latent entity representation from data. More concretely, we embed each entity into a low dimensional space. In addition, for each relation r and each argument slot, we introduce a low dimensional vector, that has the same dimension as the entity vector. For example, binary relations have two vectors, \mathbf{v}_r^1 for argument 1, and \mathbf{v}_r^2 for argument 2. Measuring compatibility of an entity tuple and relation amounts to measuring, and summing up, compatibility between each argument slot representation and the corresponding entity representation. This leads to:

$$\theta_{t,r} = \sum_c a_{t_1,c} v_{r,c}^1 + \sum_c a_{t_2,c} v_{r,c}^2$$

In this equation, t_1 stands for argument 1, likewise for argument 2.

Similarly, we can have a combined model of three parts, factorization model, neighbor model, and this entity model. This leads to:

$$\theta_{t,r} = \sum_c a_{t,c} v_{r,c} + b_r + b + \sum_i w_i f_i(r', r) + \sum_c a_{t_1,c} v_{r^1,c} + \sum_c a_{t_2,c} v_{r^2,c}$$

7.2.4 Alternative Training Objectives

To train our models, we can also employ a ranking based objective, known as Bayesian Personalized Ranking (BPR) in recommendation [Rendle et al., 2009, Krohn-Grimberghe et al., 2012]. This objective function assumes that in each row, the observed cells are positive feedback, and should be ranked ahead of the unobserved cells. Instead, our sampling based method randomly samples the unobserved cells as negative training data. In our experiments we tried both training objectives.

In universal schema, for each tuple, we can train our model to rank the observed positive cells above the negative cells [Rendle et al., 2009]. We assume each ranked pair is a random variable drawn from a Bernoulli distribution. For example, for a

tuple t , i.e. one row in the input matrix, we rank relation r over r' . This can be denoted as

$$x_{t,r} >_{\theta} x_{t,r'}$$

$$x_{t,r} = \sum_c a_{t,c} v_{c,r}$$

Similarly, here for a ranked pair, we can define the natural parameter as:

$$x_{t,r} - x_{t,r'}$$

We can learn the parameters using maximum log likelihood as well. Following the procedure in §7.2, we need to calculate

$$\frac{\partial}{\partial \theta} (x_{t,r} - x_{t,r'})$$

This falls out as

$$\frac{\partial}{\partial \theta} x_{t,r} - \frac{\partial}{\partial \theta} x_{t,r'}$$

We can also rank entity tuples for each relation if we care about the top ranked entity tuples for each relation. Here the natural parameter is

$$x_{t,r} - x_{t',r}$$

This relation based ranking is used in our evaluation §7.3.4. We can optimize the parameters using stochastic gradient descent, due to the large number of ranked pairs.

7.3 Evaluation

In this section, I discuss evaluation of our universal schema approach for relation extraction.

Our work aims to predict new relations that hold for each entity tuple. From the column perspective, this is to discover new instances for each relation. From the row perspective, this is to predict new relations for each entity pair. We currently concentrate on binary relations.

7.3.1 Data

Our approaches for relation extraction usually involve two types of data, text data and structured data. Following this, in universal schema, we use 20 years of New York Times articles [Sandhaus, 2008] as our text corpus, and Freebase [Bollacker et al., 2008] as our structured data. Freebase covers a set of entities and the relations among them. Part of the data is obtained from Wikipedia infoboxes, and part is from human annotation. More details of the text data and preprocessing can be found in §2.7.

As described in §2.7, we perform named entity recognition and dependency parsing over the NYT documents. We extract each entity pair occurring in a sentence as a candidate tuple, the dependency path that connects the two named entities as the surface pattern to denote the relation between an entity pair. This results in approximately 400,000 entity pairs and 8,000 relations.

As we mentioned many times in this thesis, many approaches discussed here require linking entities from a text corpus to a knowledge base §2.8. Here for universal schema, we perform entity linking as described in §2.8. We leave integration of universal schema with entity resolution as a future topic.

After linking entity mentions in NYT data to Freebase entities, we add Freebase relations that hold for entity pairs appearing in the text corpus. This adds 116 relations to our universal schema.

7.3.2 Evaluation Measures

The main challenge is that we do not have ground truth for the whole matrix. For example, for each entity tuple, we do not have the whole set of relations that hold. Our solutions are two-fold: one is to use the knowledge base, and the other is to use data from human annotation. When we care about rankings, we can use measures from the information retrieval (IR) community. Consider target applications, such as question answering. Usually a question can be converted to a relation, and we are interested in the top ranked entity tuples for this particular relation. We employ ranking based training objective to rank entity tuples for a relation. Our approach can rank entity pairs not only for relations defined in the knowledge base, but also for relations defined by surface patterns.

In terms of relation prediction, we can employ classification based measures. We predict relations that hold for one entity tuple and measure whether the predictions are correct. Precision, recall and F1 measures could be used. This allows us to compare against previous distant supervision based approaches. Distant supervision systems model probabilities of relations in KB conditioned on the observed surface patterns. Instead, our approach jointly models the target relation types and the surface patterns. You can see that the distant supervision approach is a discriminative model and our model is a generative model.

In our experiments, we split the data into training and test as follows. The NYT articles after 2000 are used as the training corpus, and articles from 1990 to 1999 as the test corpus. We also split Freebase facts 50/50 into train and test facts, and their corresponding tuples into train and test tuples. Train tuples are linked to training corpus, and test tuples are linked to test corpus. We evaluate our predictions for Freebase relations and sampled surface patterns.

7.3.3 Baselines

We compare different variations of our models, the neighbor model (N), the matrix factorization model (F), the combined factorization and neighbor model (NF), and the combined factorization, neighbor and entity model (NFE). To get the ranking list for each relation, we maximize the ranking based log likelihood §7.2.4.

We compare our results against distant supervision approaches for relation extraction, including DS [Mintz et al., 2009], Unsup [Yao et al., 2011], and MIML [Surdeanu et al., 2012].

7.3.4 Ranking based Evaluation

This ranking based evaluation is inspired by the TREC competitions and work in information retrieval [Manning et al., 2008]. That is, we treat each relation as a query and receive the top 1000 (run depth) entity pairs from each system. Then we pool the top 100 (pool depth) answers from each system and manually judge their relevance or “truth.” This gives a set of relevant results that we can use to calculate recall and precision measures. In particular, we can use these annotations to measure an average precision across the precision-recall curve, and an aggregate mean average precision (MAP) across all relations. This metric has shown to be robust and stable [Manning et al., 2008]. In addition, we also present a weighted version of MAP (weighted MAP) in which the average precision for each relation is weighted by the relation’s number of true facts.

Our evaluation deviates from previous work in distant supervision. Evaluation in previous work (a) combines the results from several relations in a single precision recall curve, and (b) uses held-out evaluation to measure how well the predictions match existing Freebase facts. This has some disadvantages. First, when aggregating across relations, results are often dominated by a few frequent relations, such as *containedby*, providing little information about how the models perform across the board. Second,

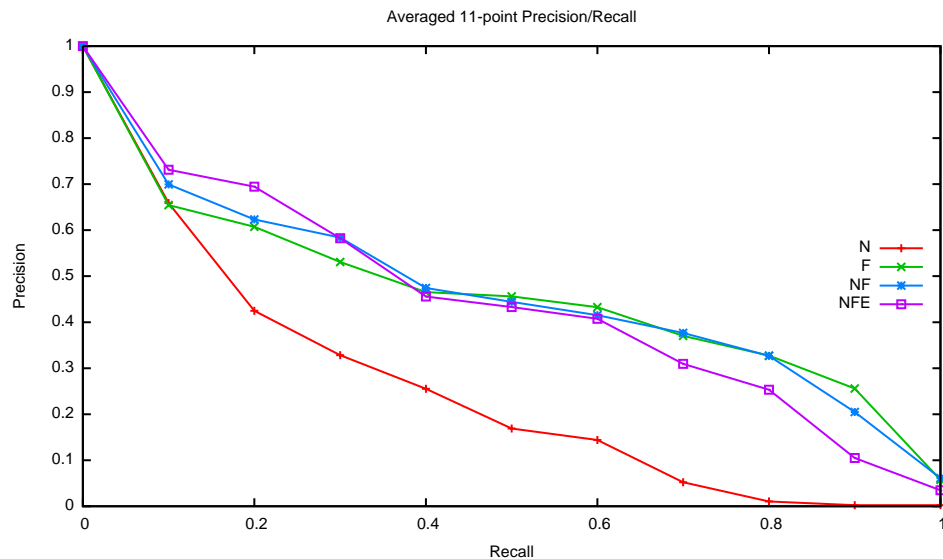


Figure 7.1. Averaged 11-point precision recall curve for surface pattern relations.

evaluating with Freebase held-out data is biased. For example, we find that frequently mentioned entity pairs are more likely to have relations in Freebase. Systems that rank such tuples higher receive higher precision than those that do not have such bias, regardless of how correct their predictions are. Our evaluation can aggregate per-relation comparisons to establish statistical significance, for example via the sign test.

Also note that while we run our models on the complete training and test set, evaluation is restricted to a subsampled test set of 10,000 tuples.

Table 7.1 summarizes our results. We can see that our approach using variations of factorizations (the last three columns) can achieve better performance than distant supervision approaches.

Figure 7.1 presents a comparison of our models with respect to 10 surface pattern relations. These relations were chosen according to what we believe are interesting questions not currently captured in Freebase. We again see that learning a latent representation (F, NF and NFE) from additional data helps quite substantially over the N model.

Table 7.1. Average and (weighted) Mean Average Precisions for Freebase relations based on pooled results. The # column shows the number of true facts in the pool. NFE is statistically different to all but NF and F according to the sign test. Bold faced are winners per relation, italics indicate ties.

Relation	#	DS	Unsup	MIML	N	F	NF	NFE
person/company	103	0.67	0.64	0.70	0.73	0.75	0.76	0.79
location/containedby	74	0.48	0.51	0.54	0.43	0.68	0.67	0.69
author/works_written	29	0.50	0.51	0.52	0.45	0.61	0.63	0.69
person/nationality	28	0.14	0.40	0.13	0.13	0.19	0.18	0.21
parent/child	19	0.14	0.25	0.62	0.46	0.76	0.78	0.76
person/place_of_death	19	0.79	0.79	0.86	0.89	0.83	0.85	0.86
person/place_of_birth	18	0.78	0.75	0.82	0.50	0.83	0.81	0.89
neighborhood/neighborhood_of	12	0.00	0.00	0.08	0.43	0.65	0.66	0.72
person/parents	7	0.24	0.27	0.58	0.56	0.53	0.58	0.39
company/founders	4	0.25	0.25	0.53	0.24	0.77	0.80	0.68
film/directed_by	4	0.06	0.15	0.25	0.09	0.26	0.26	0.30
sports_team/league	4	0.00	0.43	0.18	0.21	0.59	0.70	0.63
team/arena_stadium	3	0.00	0.06	0.06	0.03	0.08	0.09	0.08
team_owner/teams_owned	2	0.00	0.50	0.70	0.55	0.38	0.61	0.75
roadcast/area_served	2	<i>1.00</i>	0.50	<i>1.00</i>	0.58	0.58	0.83	<i>1.00</i>
structure/architect	2	0.00	0.00	<i>1.00</i>	0.27	<i>1.00</i>	<i>1.00</i>	<i>1.00</i>
composer/compositions	2	0.00	0.00	0.00	0.50	0.67	0.83	0.12
person/religion	1	0.00	<i>1.00</i>	<i>1.00</i>	0.50	<i>1.00</i>	<i>1.00</i>	<i>1.00</i>
film/produced_by	1	<i>1.00</i>	<i>1.00</i>	<i>1.00</i>	<i>1.00</i>	0.50	0.50	0.33
MAP		0.32	0.42	0.56	0.45	0.61	0.66	0.63
Weighted MAP		0.48	0.52	0.57	0.52	0.66	0.67	0.69

7.3.5 Classification based Evaluation

To evaluate Freebase predictions, we randomly sample approximately 2,000 entity pairs from test set, and obtain ground truth labels from Freebase and human annotations. We allow each entity pair to have multiple relation types. We report precision, recall and F1 on the test set. This evaluation considers the traditional relation extraction task, where predicting relations of entity pairs are important. The ranking based evaluation (§7.3.4) puts more emphasis on ranking entity pairs with respect to each relation.

The first four rows of Table 7.2 shows the performance of variations of our models. We can see that the matrix factorization model (F) performs better than the neighbor model (N). The combined model performs significantly better than the matrix factorization and the neighbor model with $p \ll 0.05$.

7.3.6 Integrating entity types

Previous work has shown that incorporating entity types can increase relation extraction accuracy [Roth and Yih, 2007, Yao et al., 2010]. Here we demonstrate this is also true in universal schema. We use universal schema to predict entity types, including Freebase entity types and surface patterns denoting unary relations. Then we use these predicted types as features in our experiments. We compare against relation extraction models based on universal schema, the matrix factorization model (F), the neighbor model (N), and the combined matrix factorization and neighbor model (F+N). Based on the combined model, we add predicted entity types as features(F+N+Unary). This is different from the entity model, where we learn low dimensional embeddings for entities and relations. Our previous experiments show that entity models do not increase the performance, so we discard entity model for this set of experiments.

System	Precision	Recall	F1
DS	0.619	0.540	0.577
N	0.591	0.611	0.601
F	0.605	0.637	0.621
F+N	0.647	0.640	0.643
F+N+NER	0.604	0.622	0.622
F+N+Unary	0.663	0.672	0.667

Table 7.2. Performance on predicting Freebase relations in universal schema. +Unary indicates adding predicted pattern based types as features. +Unary(FB) for adding predicted Freebase types

We design features as conjunctions of entity type combinations and relations, $(type1, type2, relation)$, where $type1$ ranges in all possible predicted types for the first entity argument, likewise for $type2$. For example, for cell (Gordon Bunshaft & Lever House, architect/structures_designed), predicting “Gordon Bunshaft” as an *architect*, and “Lever House” as a *location*, we add one feature (architect, location, architect/structures_designed). Intuitively, this feature type captures the selectional preferences of relations—particular relations only hold between specific argument types.

We perform experiments on the same train and test data as used in [Riedel et al. \[2013\]](#). This dataset is obtained from NYT articles and Freebase. The same annotated data set is used as ground truth for evaluation.

We allow each entity pair to have multiple relation types. We report precision, recall and F1 on the test set. Table 7.2 shows the performance of all the systems. When incorporating our entity type predictions into the combined model, F1 increases by approximately 2.5%. We also perform experiments replacing entity types with Stanford NER tags [[Finkel et al., 2005](#)], and the F1 score decreases. Comparing the neighbor model with our best model, F1 increases by 6.6%, that corresponds to a 16.5% error reduction.

Relation	Two argument types
architect/structures	designer → home
sports_team/league	team → game/competition
team_owner/teams	chairman → team
broadcast/area_served	news → suburb
military_person/conflicts	senator → movement

Table 7.3. Selectional preferences learned by our model

To better understand the effect of using entity types, we rank the features for some Freebase relations by their weights. Table 7.3 lists the top ranked type combinations for some of the relations. We can see that our model learns reasonable selectional preferences.

7.4 Exploration on Facets of Universal Schema

In this section, we explore the characteristics of universal schema on a large scale dataset, ClueWeb. ClueWeb is a data set of crawled web pages¹. We use ClueWeb12 data. We apply our NLP tools implemented in Factorie [McCallum et al., 2009] to process the web pages. We also take advantage of a corpus that automatically annotates entities from Freebase in these web pages [Gabrilovich et al., 2013]. We extract sentences that contain at least one entity mention from each page, and parse these sentences using a transition parser. For two entity mentions occurring in the same sentence, we extract the dependency path as patterns that expresses their relationship. Since the entity mentions are linked to Freebase entities, we collapse multiple mentions of the same entity pair as one row in our input matrix. We also add Freebase relation types if one pair of entities occurs in Freebase.

¹<http://www.lemurproject.org/clueweb12.php/>

#Row	#Col	#Cell	#Relation
2,690,096	9,289	5,604,691	2,036

Table 7.4. Statistics of processed data.

train	1/3	2/3	3/3
F1	0.196	0.209	0.234

Table 7.5. Performance variations of relation extraction as the amount of training data increases.

We vary different parameters, for example, the amount of training data, the number of dimensions, and the initialization methods, to test how these factors affect the performance.

7.4.1 Does more training data lead to better performance?

Here we test whether more data will increase the performance for relation extraction, i.e. predicting relation types for entity pairs. We process the ClueWeb data and collect approximately 5M entity pairs. We filter out entity pairs with only 1 pattern since our approach counts on co-occurrences to learn good embeddings. Table 7.4 shows the final statistics for the whole dataset.

We keep the most frequent 300 Freebase relations as our target labels for relation extraction. In other words, we classify the entity pairs into these 300 categories. We note that training data is limited for other Freebase labels. We split the documents into train and test. We use one third, two thirds and all of the training data. We test on the same set of held-out documents. Table 7.5 lists the performance of different amounts of training data. We can see that as training data increases, the F1 increases.

#comp.	10	50	100	200	500	1000
F1	0.061	0.212	0.243	0.250	0.261	0.259

Table 7.6. Performance variations of relation extraction as the number of dimensions varies.

0.258	0.243	0.241
-------	-------	-------

Table 7.7. Performance variations of relation extraction for different initializations.

7.4.2 Does the number of components matter?

We vary the number of components for the embedding vectors. Experiments show that in some ranges, from 100 to 500, the performance does not vary much, and a larger number of components leads to slightly better performance (see Table 7.6). The performance does not increase when increasing the number of components to 1000. Using fewer dimensions leads to worse performance. Usually we use 100 dimensions.

7.4.3 Does the non-convex objective affect the performance?

The objective is not convex. If we split the parameters into row vectors and column vectors, the objective is convex with respect to each group. Here we measure the effect of initialization on the final performance. We initialize the parameters with samples from a uniform distribution between 0 and 0.01. In different runs, we use different random seeds. As seen in Table 7.7, the performance varies with different random initializations. One can try different random initializations and average the performance. In these experiments, we use 100 components.

7.4.4 How does coreference affect the final performance?

Since we have annotations for linking Freebase entities to their mentions in ClueWeb, we can compare this entity linking against simple string match for relation extraction. We run universal schema on the same set of triples, except that we collapse

string match	entity linking
0.183	0.245

Table 7.8. Performance variations of relation extraction as we use different coreference approaches.

these triples using entity linking and string match separately. We find that the matrix created from string match is more sparse. We test on the same set of entity pairs. Table 7.8 lists the results. We can see that coreference does boost the performance. One reason why string match performs worse is that we link each entity pair to all possible Freebase relations, since one entity surface string may correspond to multiple Freebase entities. In this case, the evidence may not be sufficient to predict all these relations.

7.4.5 Can our approach discover implications among relations?

In this section, we measure the asymmetric implications among relations. For example, we can predict that “biologist” implies “scientist,” but not in the other direction. In other words, comparing against each other, “biologist” is a narrow relation and “scientist” is a broad relation. There is no standard way to measure this, so we employ different measures to uncover the mysteries of implications.

Directional similarity. To measure this asymmetric implications, we compute directional similarity for each pair of low dimensional vectors using average precision (AP) [Kotlerman et al., 2010]. This similarity measures the directional implications of two vectors, a vector \mathbf{u} of a narrow relation and a vector \mathbf{v} of a broad relation. This similarity originates from the average precision in information retrieval. Each component in \mathbf{u} is seen as a retrieved document in information retrieval, and components in \mathbf{v} correspond to a set of relevant documents. Then we rank the components of \mathbf{u} and calculate average precision. The similarity is calculated as:

implication	similarity
daughter \implies parents	0.485
parents \implies daughter	0.416
biologist \implies scientist	0.342
scientist \implies biologist	0.268
soccerteam.player \implies team.player	0.260
team.player \implies soccerteam.player	0.233
pitch for \implies be player for	0.248
be player for \implies pitch for	0.154

Table 7.9. Examples of asymmetric implication pairs.

$$\begin{aligned}
 \text{AP}(\mathbf{u} \implies \mathbf{v}) &= \frac{\sum_{c \in F(\mathbf{u})} P(c) \text{rel}(c)}{F(\mathbf{u})} \\
 P(c) &= \frac{\#\text{components before } c \text{ included in } \mathbf{v}}{\text{rank}(c, F(\mathbf{u}))} \\
 \text{rel}(c) &= 1 - \frac{\text{rank}(c, F(\mathbf{v}))}{F(\mathbf{v}) + 1} \text{ if } c \in F(\mathbf{v})
 \end{aligned}$$

where $P(c)$ is similar to precision and $\text{rel}(c)$ is a relevance score. If c does not occur in \mathbf{v} , $\text{rel}(c)$ is 0.

Since our vectors do not have components with value 0.0, we cut the absolute values of components below a threshold 0.1 as 0.0. In Table 7.9 we show some example pairs. In these pairs, one directional similarity is larger than the other. This shows that our approach captures asymmetrical implications among relations.

In order to get an overview of all the embedding vectors, we show some example broad relations and the top relations that can imply them in Table 7.10. We can see that our approach learns accurate implications.

Turned-on patterns. Besides directional similarity, we employ a method that is simple and straightforward to measure whether the embedding vectors capture

broad	narrow
person.parents	be daughter of, daughter of, son of, be son of, bear to, father, grandson, dad, descendant of, mother
place.of_birth	born in, born on #TIME# in, born on, born at, born near, native of, be originally from, grew up in, places.lived
play for	make debut for, start for, be player for, draft by, player.team, sign contract with, score for, score goal for, pitch for, add for, hit for, trade to

Table 7.10. Top narrower relations for some broader relations.

the asymmetry between two relations. Our hypothesis is that we can predict more relations based on one observed broad relation, whereas we can predict fewer relations based on observed narrow relation. We name the phenomenon of predicting relations based on observations as “turn on.” The intuition is that broad relations interact with more other relations and the embedding vectors encode these interactions. In other words, the components of these embedding vectors have more information of other relations, whereas components of narrow relation vectors have little information of other relations. At prediction time, broad relations will trigger more other relations.

After sampling tuples that have only one observed relation, we check, for each observed relation, how many relations they turn on. We show some observed relations and the number of relations they turn on in Table 7.11. Similar relations appear in one row. Each row contains one broad relation and some narrow relations. We can see that broad relations turn on more other relations, as relation “player” turns on 451 relations; whereas narrow relations turn on fewer other relations, as “pitcher” turns on 266 relations. This verified our hypothesis and show that our approach can make asymmetric predictions.

The peakiness and norm of vectors. We also employ L2 norm and L1 norm to measure the peakiness of a vector. The larger the ratio of L2 over L1 is, the

broad	narrow
politician 634	governor of 446, candidate 410, representative 366
professor 326	economist 254, physicist 243, historian 314, chemist 210
child of 447	daughter of 135, son of 188
player 443	guard 283, pitcher 270, quarterback 290

Table 7.11. The number of relations predicted as true based on only one observed relation. Broad relations (i.e., “player”) turn on more predictions; narrow relations (e.g., “pitcher”) turn on fewer predictions.

more peaked the vector is. We find that the ratios of different relations do not vary much, ranging from 0.053 to 0.059. This measure cannot tell us more information about vectors of different relations. L2 norm itself is more informative. Comparing L2 norm of the vectors, we find that broad relations have larger norms. For example, Freebase relation “location/contains” has larger norm than the Freebase relation “location/capital” and the textual pattern “capital of province.” We have similar observations for several other pairs of broad and narrow relations, including “scientist” vs “biologist,” “parent of” vs “daughter of”, and “leader of” vs “president of.”

The similarity between vector \mathbf{v}_t and \mathbf{v}_o . Here we study the similarity between a tuple vector and its single observation vector. We call the tuple vector as \mathbf{v}_t and the observation vector as \mathbf{v}_o . Our hypothesis is that for an observed narrow relation, \mathbf{v}_o is more similar to \mathbf{v}_t than an observed broad relation. We use a synthetic example to explain the intuition behind this hypothesis. Theoretically, when a tuple has one single observation, after training, \mathbf{v}_o should be close to \mathbf{v}_t since the training objective is to maximize the probability of this observation. Assume that we only have two relations and two tuples in our data set, relation n occurs with the first tuple and relation b occurs with the second tuple. After training, each of these two relation vectors (\mathbf{v}_n and \mathbf{v}_b) is similar to their corresponding tuple vector. Now we add a third tuple that has two observed relations, relation b and relation r . We retrain the model using all three tuples. The vector \mathbf{v}_b should deviate from the previous one since

relation b co-occurs with another relation r in the third tuple, whereas the vector \mathbf{v}_n should stay the same.

In our experiments, we select tuples that have only one observation for analysis. We calculate the similarity between \mathbf{v}_t and \mathbf{v}_o for each tuple. To study the similarity differences of narrow and broad relations, we average the similarities for each relation. We find that the similarities of narrow relations, especially the fine-grained ones, are higher than the similarities of broad relations. For example, the average similarity of specific relation “receive (PhD/MS/BS) degree from is 0.985, and the average similarity of the relatively broad relation “graduate of is 0.938. In another example, the similarity of “CEO of is 0.943, and the similarity of “president is 0.880. The relation “president is broad since it can represent “leader of a country, “leader of a company, and “leader of some organization. In an extreme case, the average similarity of the pattern “NNP is 0.629. This pattern represents a broad relation stating that there exists a noun phrase between the two entities of a tuple.

7.4.6 Error Analysis

In this section, we take a close look at predictions made by our matrix factorization model (F).

We mainly look at predicted instances of Freebase types since we can compare them against those predicted by other distant supervision systems. We group the errors into several categories.

Noisy training data. When building our input matrix, we link Freebase entities to their mentions in the text corpus. Each relation instance has multiple relation mentions, i.e., multiple sentences that mention the entity pair. We know that not all sentences mentioning the entity pair express the relationship between them §3.2. For example, there are few occurrences of relation expressions for some Freebase relation types, such as book/work/written_subject. This relation type may

have entity pairs like (Woodstock, New York) and (Vietnam War, Vietnam). In Freebase, there exists one piece of work named “Woodstock,” and it is “a 1970 American documentary of the watershed counterculture Woodstock Festival that took place in August 1969 at Bethel in New York.” However, in our text corpus, these two entities usually are two locations. Our system predicts few instances for relation type book/work/written_subject. Similar errors occur for relation type book/edition/place_of_publication.

Confusion among co-occurring relation types. Some relation types usually co-occur with each other and this leads to inaccurate predictions. For example, the movie “Wild Wild West” has director and producer “Barry Sonnenfeld.” This results in predicting film/directed_by, film/written_by, and film/produced_by when in fact only one of them is true.

Inaccurate dependency paths. Since we mainly employ a dependency path between two entities to represent the relation between them, sometimes incorrect path leads to wrong predictions. For example, in sentence “She was born in Morristown, the daughter of George W. Jenkins”, we incorrectly extract the path “daughter of” between “Morristown” and “George W. Jenkins.” This results in wrong prediction person/parents for (Morristown, George W. Jenkins).

Ambiguous patterns. Some patterns are ambiguous. Consider “home of” in these three clauses: “Jive Records, home of the Backstreet Boys,” “Candlestick Park, home of the San Francisco Giants,” and “Monticello, the home of Thomas Jefferson.” Our system is inclined to predict two relation types, music/record_label/artist and sports/facility/teams when observing the pattern “home of.” We do not learn the sense “home of a person” from the training data. Entity types can disambiguate these three senses.

Lower score for correct relation types. A large proportion of errors belong to this category. That is, sometimes we can predict the correct relation type with the

highest score when comparing against other types, but the absolute value is below our global threshold. For example, person/nationality ranks first for entity pair (Maya Usova, Russia), but the score is below the global threshold 0.5.

Insufficient evidence. We do not observe sufficient occurrences of some patterns. Or sometimes we observe many occurrences of these patterns, but we do not observe many co-occurrences of these patterns with other patterns or Freebase relations. Considering the entity pair (Mr. Jones, Columbia College), there exists a pattern “graduate of” between them. Ideally, we should predict the person/company relation type for this pair. However, because “graduate of” usually occurs by itself, not so often co-occurring with other patterns or Freebase types, we miss the relation type person/company. Adding more data may lead to co-occurrences of this pattern with other patterns and Freebase types.

We categorize the errors more specific to ClueWeb data as well.

Missing labels in Freebase. Freebase is not complete. Some facts predicted by our approach are indeed correct, though missing from Freebase. For example, we predict many instances for Freebase relation `fictional_universe.fictional_character.parents`, including pair (Adam/m.09_c5v, Creator/m.0f5d2). In another interesting example, we predict the sibling relationship for entity pair (Guan Yu, Liu Bei). These two persons are characters occurring in Chinese novels and they are “blood brothers.”

Incorrect entity linking. Considering entity pair (Kennedy, Rose), “Rose” is incorrectly linked to `rose(flower)`. Actually “Rose” is the first name of JF Kennedy’s mother. Our prediction `person/parents` is correct.

Short patterns. Usually short patterns are extracted for entity pairs occurring in ClueWeb. These patterns are ambiguous and are not strong indicators for particular relations. For example, patterns “at,” “of” for predicting relation `institution/parent_institution`. Entity types and more context features can help.

7.5 Related Work

Our task is similar to collaborative filtering. We only have positive data; that is, we only observe what relations hold for an entity pair. This is analogous to positive only collaborative filtering, and researchers have developed different models, including Bayesian personalized ranking [Rendle et al., 2009, Rendle and Schmidt-Thieme, 2010] and different weighing strategies [Pan and Scholz, 2009, Hu et al., 2008].

Co-clustering and matrix factorization approaches have been employed in relation extraction. Bollegala et al. [2010] employ co-clustering to find clusters of entity pairs and patterns jointly. Infinite Relational Model [Kemp et al., 2006] provides a framework to discover latent structures jointly for an n-dimensional matrix, and each dimension has a latent structure. Takamatsu et al. [2011] use probabilistic matrix factorization as a dimensionality reduction technique to discover relations. Their goals are to cluster patterns whereas our aim is to predict the source patterns.

Recently, factorization models have gained much more attention in analyzing relational data. In this section, we take several factorization models as representatives to analyze their differences, as listed in Figure 7.2. These models are different in terms of how they represent relations.

The first model represents each relation as a matrix whereas the remaining models represent each relation as a vector. The factorization of Yago [Nickel et al., 2012] is an example of the first model. There are also variations of the first model, for example, factorizing matrices that represent relations again into low dimensional vectors [Jenatton et al., 2012]. The disadvantage of the first model is that the number of parameters is large compared with representing a relation as a vector. This model is an example of factorizing tensor data.

The second model is another typical tensor factorization model [Kolda and Bader, 2009, Kang et al., 2012]. Kolda and Bader [2009] cover most of the methodologies and applications. Among them, the CANDECOMP/PARAFAC (CP) method

is mostly used for decomposing an entity/entity/relation matrix. The authors also introduce many applications for tensor decomposition. The most relevant to us are the applications in text mining and web mining. For example, researchers employ a user/query/page tensor for web page recommendation [Sun et al., 2005], and a page/page/anchor tensor for analyzing web page links [Kolda et al., 2005]. Miettinen [2011] employ boolean tensor factorizations for entity/entity/relation data. However, they work with a small data set, and mostly concentrate on exploring the algorithm with synthetic data other than using it for predictions.

The third model is similar to the second tensor factorization model. The only difference is that it uses a different way to calculate a score for each triple. The second tensor factorization model employs the tensor product of three vectors, i.e., the vectors for two entities and the vector for the relation. The third model makes the assumption that adding the first entity vector and the relation vector should lead to a vector that is close to the second entity vector [Bordes et al., 2013, Weston et al., 2013]. In other words, the relation vector translates the first entity vector into the second entity vector. The objective is to minimize the distance between the second entity vector and the sum of the first entity and relation vectors. In experiments, they show that this model can rank the first or the second entity accurately given the other two elements in a triple.

Our model is different from all the three models in that we embed each entity pair, instead of each entity, as a vector. Similar to the second and third model, we represent each relation as a vector instead of a matrix. It is advantageous to represent an entity pair as a vector over to represent each entity as a vector. Only modeling the interaction of individual entities with relations fails to capture that a relation occurs between an entity pair, not individual entities. Representing each entity as a vector breaks the interactions between two entities.

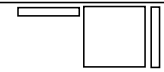

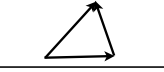
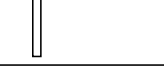
	#Parameters	References
	$e \times k + r \times k^2$	Factorizing YAGO, WWW12 Semantic Matching Energy, Machine Learning 2013
	$e \times k + r \times k$	Canonical Decomposition, Kolda and Brett W. Bader, SIAM09
	$e \times k + r \times k$	Translation model, Bordes et al., NIPS13
	$p \times k + r \times k$	Universal Schema, NAACL13

Figure 7.2. Different factorization models for relation extraction. They are different in terms of how they model relations and entities. The first model represents relations as matrices. The remaining models represent relations as vectors. All the first three models represent each entity as a vector. The fourth model, our factorization model, represents each entity pair as a vector. Except for the first model, all the models are scalable.

Our approach is also related to learning low-dimensional embeddings of high-dimensional NLP data. This topic has been of both long-standing [Brown et al., 1992, Bengio et al., 2003, Collobert and Weston, 2008, Blitzer et al., 2004] and increasing recent interest [Socher et al., 2010, Mikolov et al., 2013]. Many of these works have been for the embedding of individual words [Brown et al., 1992, Bengio et al., 2003, Blitzer et al., 2004, Mikolov et al., 2013]. There are also embeddings for structured natural language processing, such as part-of-speech tagging, phrase chunking, named entity recognition, semantic role labeling [Collobert and Weston, 2008], and parsing [Socher et al., 2010]. Our work is the first to use “open domain” universal schema as relation and entity types by leveraging natural language inputs.

7.6 Conclusion

We present universal schema for relation extraction. Universal schema contains surface patterns as relations, as well as pre-defined types from structured sources as relations. By predicting missing tuples for surface pattern relations we can populate

a knowledge base without any labelled data, and answer questions not supported by the structured schema alone. By predicting missing tuples in the structured schema we can expand a knowledge base of fixed schema; this only requires a set of existing facts from this schema. Crucially, by predicting and modeling both surface patterns and structured relations simultaneously we can improve performance. We show this experimentally by contrasting a series of the popular distant supervision models to our collaborative filtering models that learn low dimensional embeddings across surface patterns and structured relations. Moreover, our models are computationally efficient, requiring less time than comparable methods, while learning more relations.

CHAPTER 8

QUESTION ANSWERING FROM FREEBASE

A common strategy in question answering is to convert a question to a relation triple, with one or more known arguments and one missing argument. For example, “what school did Sir Ernest Rutherford go to?” would be converted to “go to (Sir Ernest Rutherford, _)” with the second argument missing. The main challenge in question answering is to understand the semantic meaning of a question, since different expressions can represent the same relation. In our example, we need to know “go to (school)” bears the same meaning as “attend (school/university/college),” and indicates Freebase relation “person.education.institution.” My research aims to discover semantic meaning expressed in natural language patterns and, if applicable, align them with pre-defined relation types. Question answering serves as a benchmark task for our work.

8.1 Question Answering System

We test our approach on question answering using Freebase, a well-known knowledge base. We do so by extracting answers from Freebase, instead of from other sources, like web pages. We choose this task since there exists a research dataset commonly used as bench mark, allowing us to compare our results with those previously published using this data. The main challenge here is to map a question to one of the relation types defined in the knowledge base, i.e., mapping “go to (school)” to the Freebase relation “person.education.institution.”

[Yao and Durme \[2014\]](#) approach this task as follows. First they parse the question, identifying the named entities, i.e., “Sir Ernest Rutherford” in our example, and note them as question topics. Second, they search question topics in the knowledge base, i.e., the Freebase entity representing “Sir Ernest Rutherford.” Third, they retrieve all the entities that are related to question topics, i.e., all Freebase entities that are related to “Sir Ernest Rutherford”, and rank these candidate entities to obtain the answers to the question. An alternative approach to question answering taken by [Bernat and Liang \[2014\]](#) is to parse a question into a logical form and execute the form on the knowledge base. For both of these approaches, we need to map a relation in a question to a relation type defined in the knowledge base, and then use this mapping to either rank candidates or to rewrite the logical forms. Because universal schemas learn embeddings for patterns and relations from knowledge bases, we can learn high quality mappings that we can evaluate using this question answering task.

To do so, we follow the approach taken by [Yao and Durme \[2014\]](#). We parse each question and extract noun phrases as named entities. We search these entities in Freebase and extract all Freebase entities related to them as candidate answers. During training, we train a classifier from questions and their answers, considering answers as positive examples and other candidates as negative examples. We use combined features from a pair of question and candidate answer. From each question, we extract entity mentions, the question word and the question focus. Usually the first word of a question is the question word. “college” is the question focus in our example. Please refer to [Yao and Durme \[2014\]](#) for definitions. For each entity mention in a question, we extract the dependency path between the question word and this entity mention. We also extract the trigger words from each question, i.e., verbs and certain nouns such as “leader,” “capital” and so on. Table 8.1 lists some example features. The first two features measure whether the candidate has the correct entity type. The third and fourth features measure selectional preferences, i.e., whether a relation

Feature	Example
qword:ctype	what:education.institution
qfocus:ctype	school:education.institution
qfocus-qtype:relation	school-person:person.institution
qtype:ctype:trigger	person:institution:go
pattern:relation	go to:person.institution
pattern-qtype:relation	go to-person:person.institution
trigger:relation	go:person.institution
trigger-qfocus:relation	go-school:person.institution
trigger-qtype:relation	go-person:person.institution
rank(relation—pattern)	rank1 (for person.institution)

Table 8.1. Combined features from question and candidates. ‘ctype’ stands for entity types of a candidate. ‘qtype’ stands for entity types of a question topic. ‘qword’ and ‘qfocus’ are similar to those defined in [Yao and Durme, 2014]

or a trigger word can take particular entity types as its arguments. The last feature measures whether a pattern in a question aligns well with a Freebase relation that answers the question. There are also some features involving triggers.

We run universal schema on ClueWeb § 7.4, using all data as training. We obtain lower dimensional embeddings for patterns and Freebase relations, and employ ranking based on the similarity between the two vectors as our features.

8.2 Experiments

We set up experiments using a question answering dataset WebQuestions [Bernat and Liang, 2014]. This dataset has a collection of questions from web suggestions and each question has annotated answers from Freebase. We split the dataset into train, development and test set as Yao and Durme [2014]. We tune parameters on the development set and report results on the test set. Table 8.2 lists the results. We can see that our F1 performance is comparable with the best result reported to date (the last row), and F1 increases when adding universal schema based features. Compared against the JHU system, our approach has higher precision but lower recall. Our

System	Prec	Rec	F1
Stanford	-	-	0.430
JHU	0.388	0.458	0.420
-UnivSchema	0.546	0.303	0.390
+UnivSchema	0.578	0.332	0.422

Table 8.2. Performance on WebQuestions: a question answering data set annotated with answers from Freebase.

lower recall is partly due to the fact that we do not have rankings for all the patterns that occur in the questions.

8.2.1 Error Analysis

While analyzing the errors our system made, we notice that some errors are caused by incorrectly parsing the questions. For example, in question “who is Jamie Little engaged to,” our system does not recognize “Jamie Little” as an entity. Some errors are due to ambiguous named entities in questions. For example, in the question “what does Janelle Brown work on,” the named entity “Janelle Brown” could be linked to several Freebase entities. In some scenarios, we fail to extract answers due to incorrect rankings or no rankings for Freebase relations. On one hand, some patterns are missing in ClueWeb data. On the other hand, some patterns are ambiguous. For example, in ClueWeb data, the pattern “be part of” often co-occurs with relation “organization.parent.child,” whereas in question answering, this pattern actually indicates relation “location.location.containedby.” Noisy training data also lead to incorrect predictions. One question in the training data asks about songs by “bob dylan.” Bob dylan has many songs, but the training data only labels his most famous song as positive.

8.3 Related Work

There are several works with results on WebQuestions data. The main differences between previous work and our own are how questions are translated to answers. [Yao and Durme \[2014\]](#) learn mappings from questions to relations by aligning relation mentions in ClueWeb data with relations from Freebase. They rank Freebase relations for each question by learning probabilities of words in questions given Freebase relations. Then they employ the ranking as features, such as rank-in-top-1, rank-in-top-3. We also use ranking based features. Instead of obtaining ranking probabilities from word given relations, we use the similarity between the embeddings of a question and a relation as our ranking criteria. One interpretation of our approach is that we learn probabilities of patterns given Freebase relations, instead of individual words.

[Bernat and Liang \[2014\]](#) learn logical derivations from question answer pairs. They translate a question into a logical formula, and execute the logical formula on Freebase to extract answers for a question. Our system does not require logical representation.

[Fader et al. \[2014\]](#) map a question to answers using a set of operators, including parsing, paraphrase, query rewrite, and execution. The process is a series of states connected by these operators. A single step is to apply an operator to a state and select a successor state since an operator can output multiple states. There are many derivations from a question to its answers and their algorithm learns to rank the derivations. Their systems does not perform better than [\[Bernat and Liang, 2014\]](#) on WebQuestions.

[Bordes et al. \[2014\]](#) embed questions and answers to a joint low dimensional space, so that vectors for correct answers are close to the vector of the question. They obtain better results on WebQuestions. Different from our approach, they use more information sources, such as the paraphrases obtained from Wikianswers.

8.4 Conclusion

We employ universal schema to learn associations of a natural language pattern to a defined relation type. Experiments demonstrate that on question answering, the associations learned by our approach, i.e., rankings of Freebase relations for a pattern, leads to better performance than several baseline systems.

CHAPTER 9

CONCLUSIONS AND FUTURE DIRECTIONS

Relation extraction plays an important role in information extraction, question answering and many other natural language tasks. In this thesis, we represent natural language patterns as syntactic dependency paths and explore several approaches for inducing the semantic meanings of these relational patterns. We apply distant supervision to align relational patterns with pre-defined relation types, employ generative models to discover semantic relation clusters that are not defined in knowledge bases, and we present universal schema to represent both natural language patterns and relation types in knowledge base. In universal schema, we employ matrix factorization to learn semantic associations among patterns and relation types.

There is increasing interest in interpreting natural languages that express relations between entities. Here we briefly list some challenging future directions.

Incorporate constraints. Assume that we know both *Seahawks* and *Patriots* are football teams, we can constrain that the low dimensional vectors for these two entities are close to each other. In modeling entities using universal schema, our objective function is:

$$\prod_{e,r} p_{\theta}(x_{e,r}) - \Lambda_{\Theta} \|\Theta\|^2$$

We can add one constraint term to the objective function:

$$\prod_{e,r} p_{\theta}(x_{e,r}) - \Lambda_{\Theta} \|\Theta\|^2 - \sum_{i,j} w_{i,j} \text{dist}(e_i, e_j)$$

In this equation, we add the constraints that e_i and e_j should be close in the low dimensional space. We can use any distance function between the vectors for two

entities. There is recent work on applying constraints to embeddings [Rocktäschel et al., 2014], where they encourage embeddings to be consistent with facts and first order rules in a knowledge base.

Embeddings in low dimensional space. Many factorization models and deep learning models aim to embed entities and relations into low dimensional space. These low dimensional vectors can help build relationships between entities, identify paraphrases for relational expressions, answer natural language questions. There are many interesting future directions to pursue, for example, exploring new embedding models, applying learned embeddings to downstream tasks, and interpreting these embeddings.

Use knowledge base in applications, for example, question answering. Answering questions by extracting information from Freebase is one simplified version of the question answering task. We can still employ our system discussed in Chapter 8 to tackle question answering. One component needs to be modified is how to extract candidate answers. With Freebase, the candidates are limited to entities in knowledge bases. In question answering, we must extract candidates from data sources. We can use information retrieval to extract the related passages first and extract candidate entities from the related passages. Assume we have one candidate from one passage, to rank the candidate, we need to extract a relation between this candidate and an entity from the question. This relation is a substitute for the Freebase relation in our system. In practice, this relation is usually not a pre-defined relation type, but patterns or word sequences extracted from the candidate passage. Our challenge is still to determine whether the relation extracted from the passage is a match for the relation in question. Questions are part of the queries submitted to search engines. Knowledge base can also help queries of entities. We can search the entity in the knowledge base, extract the other entities that are related to the query entity, and present the structured results to users.

Data integration. Universal schema can serve as a framework for various data integration tasks. For example, we could integrate facts from one schema (say, Freebase) into another (say, the TACKBP schema¹) by adding both sets of relations to the set of surface patterns. Reasoning with this schema will mean populating each knowledge base with facts from the other, and would leverage information in surface patterns to improve integration.

¹<http://www.nist.gov/tac/2014/KBP/>

BIBLIOGRAPHY

- Amit Bagga and Breck Baldwin. Algorithms for scoring coreference chains. In *The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, 1998.
- Pierre Baldi, Søren Brunak, Yves Chauvin, Claus A. F. Andersen, and Henrik Nielsen. Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, 16:412–424, 2000.
- Krisztian Balog and Robert Neumayer. Hierarchical target type identification for entity-oriented queries. In *Proceedings of CIKM*, 2012.
- Michele Banko and Oren Etzioni. The tradeoffs between open and traditional relation extraction. In *Proceedings of ACL-08: HLT*, 2008.
- Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open information extraction from the web. In *Proceedings of IJCAI2007*, 2007.
- Kedar Bellare, Partha Pratim Talukdar, Giridhar Kumaran, Fernando Pereira, Mark Liberman, Andrew McCallum, and Mark Dredze. Lightly-supervised attribute extraction for web search. In *NIPS workshop on Machine Learning for Web Search*, 2007.
- Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.
- Jonathan Bernat and Percy Liang. Semantic parsing via paraphrasing. In *Proceedings of ACL*, 2014.
- David Blei, Andrew Ng, and Michael Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003.
- John Blitzer, Kilian Q. Weinberger, Lawrence K. Saul, and Fernando C. N. Pereira. Hierarchical distributed representations for statistical language modeling. In *Proceedings of NIPS*, 2004.
- Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-102-6. doi: <http://doi.acm.org/10.1145/1376616.1376746>.

- Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. Relational duality: Unsupervised extraction of semantic relations between entities on the web. In *Proceedings of WWW*, 2010.
- Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of NIPS*, 2013.
- Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. In *Proceedings of EMNLP*, 2014.
- Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.
- Razvan C. Bunescu and Raymond J. Mooney. Learning to extract relations from the web using minimal supervision. In *ACL*, 2007.
- Andrew Carlson, Justin Betteridge, Richard Wang, Estevam Hruschka, and Tom Mitchell. Coupled semi-supervised learning for information extraction. In *Third ACM International Conference on Web Search and Data Mining (WSDM '10)*, 2010.
- Richard A. Caruana. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of ICML*, 1993.
- Jonathan Chang, Jordan Boyd-Graber, Chong Wang, Sean Gerrish, and David Blei. Reading tea leaves: How humans interpret topic models. In *Proceedings of NIPS*, 2009.
- Eugene Charniak and Micha Elsner. Em works for pronoun anaphora resolution. In *Proceedings of ACL*, 2009.
- Harr Chen, Edward Benson, Tahira Naseem, and Regina Barzilay. In-domain relation discovery with meta-constraints via posterior regularization. In *Proceedings of ACL*, 2011.
- Jackie Chi Kit Cheung and Xiao Li. Sequence clustering and labeling for unsupervised query intent discovery. In *Proceedings of WSDM*, 2012.
- Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical methods in natural language processing (EMNLP '02)*, volume 10, pages 1–8, 2002.
- Michael Collins, Sanjoy Dasgupta, and Robert E. Schapire. A generalization of principal component analysis to the exponential family. In *Proceedings of NIPS*, 2001.

- Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*, 2008.
- Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003. ISSN 1533-7928.
- M. Craven and J. Kumlien. Constructing biological knowledge-bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 77–86, Germany, 1999.
- Aron Culotta and Jeffery Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of ACL*, Barcelona, Spain, 2004. URL <http://www.cs.umass.edu/~culotta/pubs/tkernel.pdf>.
- Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, 2005.
- Asif Ekbal, Eva Sourjikova, Anette Frank, and Simone Paolo Ponzetto. Assessing the challenge of fine-grained named entity recognition and classification. In *Named Entities Workshop, ACL*, 2010.
- Micha Elsner, Eugene Charniak, and Mark Johnson. Structured generative models for unsupervised named-entity clustering. In *Proceedings of NAACL*, 2009.
- Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Open question answering over curated and extracted knowledge bases. In *Proceedings of KDD*, 2014.
- Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.
- Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL '05)*, pages 363–370, June 2005.
- Michael Fleischman and Eduard Hovy. Fine grained classification of named entities. In *Proceedings of Coling*, 2002.
- Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. FACC1: Free-base annotation of ClueWeb corpora. 2013.
- S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. pages 452–472, 1990.
- Claudio Giuliano and Alfio Gliozzo. Instance-based ontology population exploiting named-entity substitution. In *Proceedings of Coling*, 2008.

- Benjamin Hachey. *Towards Generic Relation Extraction*. PhD thesis, University of Edinburgh, 2009.
- Aria Haghighi and Dan Klein. Coreference resolution in a modular, entity-centered model. In *Proceedings of HLT-NAACL*, 2010.
- Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. Discovering relations among named entities from large corpora. In *Proceedings of ACL*, 2004.
- Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 2012.
- Raphael Hoffmann, Congle Zhang, and Daniel S. Weld. Learning 5000 relational extractors. In *ACL*, 2010.
- Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL*, 2011.
- Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of ICDM*, 2008.
- Rodolphe Jenatton, Nicolas Le Roux, Antoine Bordes, and Guillaume Obozinski. A latent factor model for highly multi-relational data. In *Proceedings of NIPS*, 2012.
- Nanda Kambhatla. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of ACL*, 2004.
- U Kang, Evangelos Papalexakis, Abhay Harpale, and Christos Faloutsos. Gigatensor: Scaling tensor analysis up by 100 times - algorithms and discoveries. In *Proceedings of KDD*, 2012.
- Rohit J. Kate and Raymond J. Mooney. Joint entity and relation extraction using card-pyramid parsing. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL' 10)*, 2010.
- Charles Kemp, Joshua B. Tenenbaum, and Thomas L. Griffiths. Learning systems of concepts with an infinite relational model. In *Proceedings of AAAI*, 2006.
- Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51:455–500, 2009.
- Tamara G. Kolda, Brett W. Bader, and Joseph P. Kenny. Higher-order web link analysis using multilinear algebra. In *Proceedings of ICDM*, 2005.
- Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 426–434, New York, NY,

- USA, 2008. ACM. ISBN 978-1-60558-193-4. doi: 10.1145/1401890.1401944. URL <http://doi.acm.org/10.1145/1401890.1401944>.
- Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-geffet. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(4):359–389, 2010.
- Zornitsa Kozareva and Eduard Hovy. Learning arguments and supertypes of semantic relations using recursive patterns. In *Proceedings of ACL 10*, 2010.
- Artus Krohn-Grimberghe, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. Multi-relational matrix factorization using bayesian personalized ranking for social network data. In *Proceedings of WSDM*, 2012.
- John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML' 01)*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.
- Dekang Lin. Dependency-based evaluation of minipar. In *Proceedings of the Workshop on the Evaluation of Parsing Systems*, 1998.
- Dekang Lin and Patrick Pantel. DIRT - Discovery of Inference Rules from Text. In *Proceedings of KDD*, 2001.
- Xiao Ling and Daniel S. Weld. Fine-grained entity recognition. In *Proceedings of AAAI*, 2012.
- Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008. ISBN 978-0-521-86571-5. URL <http://nlp.stanford.edu/IR-book/information-retrieval-book.html>.
- Andrew McCallum, Karl Schultz, and Sameer Singh. Factorie: Probabilistic programming via imperatively defined factor graphs. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1249–1257. 2009.
- Pauli Miettinen. Boolean tensor factorizations. In *Proceedings of ICDM*, 2011.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representation in vector space. In *Proceedings of workshop at ICLR*, 2013.
- Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of NAACL*, 2013.
- Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *ACL-IJCNLP*, 2009.

- Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing yago: Scalable machine learning for linked data. In *Proceedings of WWW*, 2012.
- J. Nivre, J. Hall, and J. Nilsson. Memory-based dependency parsing. In *Proceedings of CoNLL*, pages 49–56, 2004.
- Rong Pan and Martin Scholz. Mind the gaps: Weighting the unknown in large-scale one-class collaborative filtering. In *Proceedings of SIGKDD*, 2009.
- Patrick Pantel, Rahul Bhagat, Bonaventura Coppola, Timothy Chklovski, and Eduard Hovy. ISP: Learning Inferential Selectional Preferences. In *Proceedings of NAACL HLT*, 2007.
- Patrick Pantel, Thomas Lin, and Michael Gamon. Mining entity types from query logs via user intent modeling. In *Proceedings of ACL*, 2012.
- Hoifung Poon and Pedro Domingos. Unsupervised semantic parsing. In *Proceedings of the Conference on Empirical methods in natural language processing (EMNLP)*, 2008.
- Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to Wikipedia. In *Proceedings of ACL*, 2011.
- Joseph Reisinger and Raymond J. Mooney. Cross-cutting models of lexical semantics. In *Proceedings of EMNLP*, 2011.
- Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of WSDM*, 2010.
- Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of UAI*, 2009.
- Sebastian Riedel, Hong-Woo Chun, Toshihisa Takagi, and Jun’ichi Tsujii. A markov logic approach to bio-molecular event extraction. In *Proceedings of the Natural Language Processing in Biomedicine NAACL 2009 Workshop (BioNLP ’09)*, pages 41–49, 2009. URL <http://www.aclweb.org/anthology/W/W09/W09-1406.pdf>.
- Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling relations and their mentions without labelled data. In *Proceedings of ECML/PKDD*, 2010.
- Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin Marlin. Relation extraction with matrix factorization and universal schemas. In *Proceedings of NAACL*, 2013.
- Bryan Rink and Sanda Harabagiu. A generative model for unsupervised discovery of relations and argument classes from clinical texts. In *Proceedings of EMNLP*, 2011.
- Alan Ritter, Mausam, and Oren Etzioni. A Latent Dirichlet Allocation method for Selectional Preferences. In *Proceedings of ACL10*, 2010.

- Alan Ritter, Luke Zettlemoyer, Mausam, and Oren Etzioni. Modeling missing data in distant supervision for information extraction. *TACL*, 2013.
- Tim Rocktäschel, Matko Bosnjak, Sameer Singh, and Sebastian Riedel. Low-dimensional embeddings of logic. In *ACL Workshop on Semantic Parsing (SP2014)*, 2014.
- Dan Roth and Wen-tau Yih. Global inference for entity and relation identification via a linear programming formulation. 2007.
- Evan Sandhaus. *The New York Times Annotated Corpus*. Linguistic Data Consortium, Philadelphia, 2008.
- Stefan Schoenmackers, Oren Etzioni, and Daniel S. Weld. Scaling textual inference to the web. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 79–88, Morristown, NJ, USA, 2008. Association for Computational Linguistics.
- Diarmuid O Seaghdha. Latent variable models of selectional preference. In *Proceedings of ACL 10*, 2010.
- Sameer Singh, Karl Schultz, and Andrew McCallum. Bi-directional joint inference for entity resolution and segmentation using imperatively-defined factor graphs. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, pages 414–429, 2009.
- Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. WikiLinks: Large-scale cross-document coreference corpus labeled via links to wikipedia. Technical Report UM-CS-2012-015, University of Massachusetts, Amherst, 2012.
- Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *NIPS*, 2010.
- Jian-Tao Sun, Hua-Jun Zeng, Huan Liu, Yuchang Lu, and Zheng Chen. Cubesvd: A novel approach to personalized web search. In *Proceedings of WWW*, 2005.
- Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. Multi-instance multi-label learning for relation extraction. In *Proceedings of EMNLP-CoNLL*, 2012.
- Idan Szpektor and Ido Dagan. Learning entailment rules for unary templates. In *Proceedings of Coling*, 2008.
- Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. Probabilistic matrix factorization leveraging contexts for unsupervised relation discovery. In *Proceedings of PAKDD*, 2011.

- Hristo Tanev and Bernardo Magnini. Weakly supervised approaches for ontology population. In *Proceedings of 11st Conference of the European Chapter of the Association for Computational Linguistics*, 2006.
- Daniel S. Weld, Raphael Hoffmann, and Fei Wu. Using wikipedia to bootstrap open information extraction. In *ACM SIGMOD Record*, 2009.
- Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of EMNLP*, 2013.
- Michael Wick, Khashayar Rohanimanesh, Aron Culotta, and Andrew McCallum. Samplerank: Learning preferences from atomic gradients. In *Neural Information Processing Systems (NIPS), Workshop on Advances in Ranking*, 2009.
- Limin Yao, David Mimno, and Andrew McCallum. Efficient methods for topic model inference on streaming document collections. In *Proceedings of SIGKDD*, 2009.
- Limin Yao, Sebastian Riedel, and Andrew McCallum. Collective cross-document relation extraction without labelled data. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1013–1023, Cambridge, MA, October 2010. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/D/D10/D10-1099>.
- Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. Structured relation discovery using generative models. In *Proceedings of EMNLP*, 2011.
- Limin Yao, Sebastian Riedel, and Andrew McCallum. Probabilistic databases of universal schema. In *Proceedings of the AKBC-WEKEX Workshop at NAACL*, 2012a.
- Limin Yao, Sebastian Riedel, and Andrew McCallum. Unsupervised relation discovery with sense disambiguation. In *Proceedings of ACL*, 2012b.
- Limin Yao, Sebastian Riedel, and Andrew McCallum. Universal schema for entity type prediction. In *Proceedings of AKBC workshop at CIKM*, 2013.
- Xuchen Yao and Benjamin Van Durme. Information extraction over structured data: question answering with Freebase. In *Proceedings of ACL*, 2014.
- Alexander Yates and Oren Etzioni. Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research*, 34: 255–296, 2009.
- Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. Hyena: Hierarchical type classification for entity names. In *Proceedings of Coling*, 2012.
- Dimitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *JMLR*, 3(6):1083 – 1106, 2003.