Exploring Relational Features and Learning under Distant Supervision for Information Extraction Tasks

submitted in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

of the
Indian Institute of Technology Bombay, India
and
Monash University, Australia
by

Ajay Nagesh

Supervisors:

Prof. Ganesh Ramakrishnan (IIT Bombay)

Dr. Gholamreza Haffari (Monash University)

Prof. Pushpak Bhattacharyya (IIT Bombay)

Prof. Geoff Webb (Monash University)

The course of study for this award was developed jointly by the Indian Institute of Technology Bombay, India and Monash University, Australia and given academic recognition by each of them. The programme was administered by The IITB-Monash Research Academy.

2015

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute/the Academy and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Place	Signature
Date	Name

Acknowledgments

Before starting my PhD program, I had come across a statement to the effect that *"Earning a PhD is like running in a marathon, except that you are not certain what the goal will look like!"* Having reached the final stages of this long race, nay, journey! this statement certainly rings true. It is a long personal journey which tests the candidate's patience, grit, determination and commitment in the face of odds and uncertainty. It is a unique personal experience, which is deeply enriching and truly humbling. Undertaking this personal journey is not an easy task, unless one is endowed with guiding lights and fellow travelers.

I have been fortunate enough to have the benefit of multiple guiding lights to make my PhD journey a rich and a unique learning experience.

First among them is *Prof. Ganesh Ramakrishnan*. He has been my mentor from the very beginning. He has not only been my advisor on technical aspects but also has been my mentor in soft skills and overall progress. Our interests not only resonated in the technical aspects but also in other areas ranging from philosophy to literature, making it truly a great collaboration. A bundle of infinite energy, a guru in the true sense and a wonderful person, I am extremely indebted to him.

Dr. Gholamreza Haffari, or, *Reza*, as he is affectionately called by collaborators and students alike, has been a wonderful advisor. I started collaborating with Reza roughly half way into the program, when I visited Monash. I have perhaps never seen a person more patient and composed than him. An amazing listener and a great mentor, his contribution towards my overall progress is invaluable.

My association with *Prof. Pushpak Bhattacharyya* has been since the days of my Master's project. Counted among the top researchers in natural language processing across the globe, I have learnt a lot about academia under his tutelage.

Prof. Geoff Webb, a pioneer in the area of Machine Learning and Data Mining,

was kind enough to mentor me when I initially approached him with a thesis proposal. Although it was not exactly overlapping with his areas of interest, he has always been encouraging me to pursue mine. His suggestions from time to time, including the one to have Reza mentor me, have been extremely helpful.

I was fortunate to have *Prof. Saketha Nath, Prof. Shonali Krishnaswamy* and *Dr. Mark Carman* as members of my Research Progress Committee. Their inputs and critical evaluation during the annual progress seminars have been very useful in shaping my thesis.

I cut my teeth in research with a collaborative project with *IBM Research, Almaden*. Our collaborators were *Laura Chiticariu* and *Rajasekar Krishnamurthy*. Both of them are not only amazing researchers but excellent mentors. In spite of being separated by multiple time zones, we could do some amazing work and I thank them for their interest and time to have innumerable number of calls during the course of the project. I would also like to thank my friend *Ankush* for all his support during this project.

Although the PhD journey is a unique one, it is immensely important to have other fellow travelers in various stages of their journey to talk to, get advise from and act as *"agony aunts"*. My good fortune is that I have been blessed with a wonderful set of people who acted as my "friends, philosophers and guides", to use the clichéd term. *Kasiviswanatha Sarma*, *Mitesh Khapra*, *Manoj Chinnakotla* and *Anoop Kunchukuttan*. If not for their wonderful words of encouragement, advice, pep talk and friendship, I would not have reached this stage. I am eternally indebted to them.

I have had an excellent peer group that made my entire stay something to cherish and reminisce throughout my life. My lab-mates in KReSIT: *Naveen Nair*, *Ramkumar Rajendran* and *Pratik Jawanpuria*. Friends from our informal Kannada group, namely, *Karthik Ramachandra, Prasannna Kumar, Abhisekh Sankaran, Vishal Anwekar, Nanditha Rao, Russell Quadros, Srinivas Karthik, Venkataswamy, Rajath Bhat, Sanjay Kumar* and *Ramakrishna Bairi*. Members of the TCS lab: *Balamurali, Manish Shrivastava, Smriti Singh, Abhijit Mishra, Vasu, Aditya Joshi, Laxmi Madam, Jaya Madam, Gajananji* and *Deepakji*. Friends who made my stay in Australia memorable: *Sushrut, Geetanjali, Chetana, Arun Konagurthu, Anna Martinez, Nayyar Zaidi, Shenglie Chen, Sai Theja Ranuva* and *Pratham Arora*.

Sincere thanks to the IITB-Monash Research Academy, former CEO *Mohan Krishnamoorty* and the entire academy staff including *Kuheli Banerjee, Kiran More, Jayasree*

T S, Anasuya Banerji, Mamata Bhattacharya, Bharat and *Rahul* for their able support during the entire program. I am especially thankful to *Sheba Sanjay*, the academy communications consultant for her meticulous work of suggesting various typographical and stylistic corrections in my thesis. Additionally my gratitude extends to the CSE Department office staff *Vijay Ambre, Mrs. Athavankar, Homcy* and *Sunanda*.

Sincere thanks to *Divakar* and *Vishvendra* for hosting me on my short trips to Mumbai. *Mr. Ramesh C. S.* and his family have been my local guardians in Mumbai, whom I used to visit when I had a craving for home-food. Prof. Ganesh's wife *Aarti Madam*, and his family members have invited me on various occasions and have been very dear to me.

My teachers from high school and college have always encouraged me to pursue academic interests and motivated me to have bigger goals. I would like to especially thank *Prof. K. S. Kannan* for introducing me to the area of Computational Linguistics and *Mrs. Kokila Amarnath* for motivating me at an early age.

My parents have been my strength throughout this journey. My dad has been a pillar of support for me all though my life. He has never imposed his ambitions on me and allowed me to realize my true potential. Thanks *Appa*! Perhaps without my mother's insistence, I would not have joined a PhD program. Having understood my interests, she has been very supportive of me, in spite of having to stay away from her dear son. Thanks a lot *Amba*! *Pooja*, has been the love of my life. My dear wife, a true partner, she has been my tower of strength at all times. Without her, this would not have been possible. Thanks my love!

Last but not the least, I would like to thank almighty, the Goddess of Light for making me who I am and constantly kindling my life force. *Om Tat Sat.*

Date: _____ Ajay Nagesh

Abstract

Information Extraction (IE) has become an indispensable tool in our quest to handle the data deluge of the information age. IE can broadly be decomposed into Named-entity Recognition (NER) and Relation Extraction (RE). In this thesis, we view the task of IE as finding patterns in unstructured data, which can either take the form of features and/or be specified by constraints. In NER, we study the categorization of complex relational[1] features and outline methods to learn feature combinations though induction. We demonstrate the efficacy of induction techniques in learning : i) rules for the identification of named entities in text — the novelty is the application of induction techniques to learn in a very expressive declarative rule language ii) a richer sequence labeling model — enabling optimal learning of discriminative features. In RE, our investigations are in the paradigm of *distant supervision*, which facilitates the creation of large, albeit noisy training data. We devise an inference framework in which constraints can be easily specified in learning relation extractors. In addition, we reformulate the learning objective in a max-margin framework. To the best of our knowledge, our formulation is the first to optimize multi-variate non-linear performance measures such as F_β for a latent variable structure prediction task.

[1]Terminology is borrowed from *logic*, where relational logic is more powerful than propositional logic with the inclusion of quantifiers, but is a subset of first-order logic

Publications from the Thesis

Peer-reviewed Conferences

1. **Ajay Nagesh**, Ganesh Ramakrishnan, Laura Chiticariu, Rajasekar Krishnamurthy, Ankush Dharkar, Pushpak Bhattacharyya. **Towards Efficient Named-entity Rule Induction for Customizability.** In *Conference on Empirical Methods in Natural Language Processing and Natural Language Learning **(EMNLP-CoNLL)**, Jeju, Korea, 2012.*

2. **Ajay Nagesh**, Gholamreza Haffari, Ganesh Ramakrishnan. **Noisy Or-based model for Relation Extraction using Distant Supervision.** In *Conference on Empirical Methods in Natural Language Processing **(EMNLP)**, Doha, Qatar, 2014.*

3. Gholamreza Haffari, **Ajay Nagesh**, Ganesh Ramakrishnan. **Optimizing Multivariate Performance Measures for Learning Relation Extraction Models.** In *Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies **(NAACL-HLT)**, Denver, USA, 2015.*

4. **Ajay Nagesh**. **Exploring Relational Features and Learning under Distant Supervision for Information Extraction Tasks.** Thesis Proposal, Student Research Workshop. In *Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies **(NAACL-HLT)**, Denver, USA, 2015.*

5. **Ajay Nagesh**, Naveen Nair and Ganesh Ramakrishnan. **Comparison between Explicit Learning and Implicit Modeling of Relational Features in Structured Output Spaces**. In *23rd International Conference on Inductive Logic Programming **(ILP)**, Rio de Janeiro, Brazil, 2013.*

6. Naveen Nair, **Ajay Nagesh**, Ganesh Ramakrishnan. **Probing the Space of Optimal Markov Logic Networks for Sequence Labeling.** In *22nd International Conference on Inductive Logic Programming **(ILP)**, Dubrovnik, Croatia, 2012.*

Journal

1. Naveen Nair, Ajay Nagesh and Ganesh Ramakrishnan, **Learning Discriminative Relational Features for Sequence Labeling**, To be communicated.

Invited Papers and Talks

1. *Towards efficient named-entity rule induction for customizability.*
   - Invited poster in the *18th International Conference on Management of Data **(COMAD)***, Pune, India, 2012.
   - Poster presentation at **IBM I-CARE**, Bangalore, India, 2012.
   - Invited paper in the *23rd International Conference on Inductive Logic Programming **(ILP)***, Rio de Janeiro, Brazil, 2013.
2. *Noisy Or-based model for Relation Extraction using Distant Supervision.*
   Poster presentation at Xerox India Research symposium (**XRCI Open**), Bangalore, India, 2015.
3. *Exploring Relational Features and Learning under Distant Supervision for Information Extraction Tasks.* In the *9th Inter-Research-Institute Student Seminar in Computer Science **(IRISS 2015)***, collocated with **ACM India Annual Event**, Goa, India, 2015.

Awards

1. Awarded a full Student Travel Grant to *EMNLP-CoNLL, Jeju, Korea, 2012.*
2. Secured an Attendance Grant to *EMNLP, Doha, Qatar, 2014.*
3. Awarded the *National Science Foundation (NSF)* grant to attend the Student Research Workshop at *NAACL, Denver, USA, 2015.*
4. Secured grants from *ACM-IARCS*, *Xerox Research Center India (XRCI)* and *Microsoft Research India (MSRI)* to travel to *NAACL, Denver, USA, 2015.*

Contents

Page

Abstract **v**

List of Tables **xi**

List of Figures **xii**

1 Introduction **1**

1.1 Information Extraction : Challenges 1

1.2 Overview of Thesis Contributions 3

1.3 Learning for Named-Entity Extraction 5

1.3.1 Feature Induction in a Rule-based Setting 7

1.3.2 Feature Induction in a Max-margin Setting 9

1.4 Learning for Relation Extraction 10

1.4.1 Relaxed Distant Supervision 10

1.4.2 Distant Supervision in a Max-margin Setting 11

1.5 Thesis Organization 12

I Learning for Named Entity Recognition **14**

2 Relational Feature Space **15**

2.1 Feature Classes 17

3 Relational Feature Induction in a Rule-based Setting **23**

3.1 Entity Recognition : A Rule-based Perspective 23

3.1.1 Building Blocks of NER rules 25

3.1.2 Annotator Development Lifecycle 25

3.1.3 Problem Statement and Research Contributions 27
3.1.4 Comparison to Related Work in Induction 27
3.2 SystemT and AQL 28
3.3 Extractor Complexity 29
3.4 Methodology : Process of Rule Induction 30
3.4.1 Basic Features and Background Knowledge 31
3.4.2 Induction of Candidate Definition Features 33
3.4.3 Induction of Candidate-Refinement Features 36
3.5 Experiments 39
3.5.1 Experimental Setup 39
3.5.2 Experiments and Results 41
3.5.3 Discussion 44
3.6 Relational Features in a Max-margin Setting 45
3.6.1 Sequence Labeling 45
3.6.2 Models for Sequence Labeling 46
3.6.3 Experiments 49
3.7 Chapter Summary 51

II Learning for Relation Extraction 53

4 Relaxed Distant Supervision 54
4.1 Relation Extraction through Distant Supervision 54
4.1.1 Related Work 55
4.2 Modeling Constraints in Distant Supervision 58
4.3 Inference via Integer Linear Programming Framework 61
4.4 Experiments 64
4.4.1 Datasets and Evaluation 64
4.4.2 Experimental Results and Discussion 65
4.5 Chapter Summary 67

5 Distant Supervision in a Max-margin Setting 70
5.1 Latent Variable Structure Prediction Tasks 70
5.1.1 Related Work 71

5.1.2 Our Contributions 71
5.2 Max-margin Formulation 72
5.2.1 Distant Supervision as a Large-margin Problem 72
5.2.2 Introduction to Structured Prediction Learning 74
5.3 Optimizing Multi-variate Performance Measures 75
5.3.1 Concave-Convex Procedure (CCCP) 75
5.3.2 Loss-Augmented Inference 76
5.3.3 Effective Optimization of the Dual 78
5.4 Experiments 80
5.4.1 Experimental Setup 80
5.4.2 Training on Sub-samples of Data 81
5.4.3 The Overall Results 85
5.4.4 Discussion 85
5.5 Chapter Summary 86

6 Conclusion **88**

Appendix **91**

A Relational Feature Class Hierarchy **91**

B SystemT and AQL **93**

C Induction Target Language **97**

Bibliography **99**

List of Tables

Table Page

3.1 Accuracies of rule-based NER systems 24
3.2 Dataset details 39
3.3 Results on CoNLL03 dataset with different basic feature sets. 40
3.4 Comparison of induced rules (with and without bias) and manually developed rules. (CoNLL03 test dataset) 43
3.5 Induced Features in Max-margin algorithm 50

4.1 Highest F1 point in P/R curve: KBP Dataset 65
4.2 Highest F1 point in P/R curve: Riedel Dataset 68

5.1 Average results on 10% Riedel datasets. 82
5.2 Local vs. Exhaustive Search. 84
5.3 Overall results on the *positive* dataset. 85
5.4 Increasing weight on Precision in F_β. 85
5.5 F_1-scores on the entire test set. 86

C.1 Phases in induction, the language constructs invoked in each phase, the prescriptions for inducing rules in the phase, and the corresponding type of rule in manual rule-development. 98

List of Figures

Figure Page

1.1 Text Snippets and Facts Extracted 2
1.2 Patterns as Feature Combinations 3
1.3 Patterns as Constraints 4
1.4 Phases of Rule Induction 7
1.5 RLGG of an example pair 8
1.6 Graphical model for Distantly Supervised Relation Extraction. 11
1.7 Thesis Organization 13

2.1 Types of Predicates 16
2.2 Simple Conjunct ($\mathcal{SC}$) 18
2.3 Absolute Feature ($\mathcal{AF}$) 19
2.4 Primary Feature ($\mathcal{PF}$) 19
2.5 Composite Feature ($\mathcal{CF}$) 20
2.6 Candidate Refinement Feature ($\mathcal{CR}$) 20
2.7 Hierarchy of Features $\mathcal{PF} \subset \mathcal{AF} \subset \mathcal{CF}$ $\mathcal{SC} \subset \mathcal{CF}$ $\mathcal{SC} + \mathcal{PF} + \mathcal{AF} + \mathcal{CF} = \mathcal{CD}$ 21
2.8 Thesis Organization: Chapter 2 22

3.1 Annotator Development Lifecycle 26
3.2 Correspondence between Manual Rule development and Rule Induction. . . 31
3.3 System Diagram with details and illustrative examples. 32
3.4 Background Knowledge Creation 33
3.5 Clustering of Examples 34
3.6 Relative Least General Generalization 35
3.7 Iterative Clustering 36

3.8 Span-View Table 37
3.9 NER as Sequence Labeling. 46
3.10 Thesis Organization: Chapter 3 52

4.1 Distant Supervision for Relation Extraction 55
4.2 Graphical model instantiated for an entity-pair: Hoffmann et al. (2011) . . 59
4.3 Statistics of datasets from Surdeanu et al. (2012) 64
4.4 Results : KBP dataset 66
4.5 Results : Riedel dataset 67
4.6 Thesis Organization: Chapter 4 69

5.1 Graphical model instantiated for an entity-pair 73
5.2 Experiments on 10% Riedel datasets. 82
5.3 Weighting of Precision and Recall 83
5.4 Overall accuracies Riedel dataset 84
5.5 Thesis Organization: Chapter 5 87

6.1 Thesis Summary 89

B.1 Example *Person*extractor in AQL 94
B.2 Output of *Person*extractor on a sample document snippet 95

Chapter 1

Introduction

Most of the content that we come across in the digital media in the form of emails, blogs, webpages, enterprise data and so on are authored in natural language and have very little structure. With the dawn of the information age, we produce a colossal amount of unstructured data every day. This presents an enormous challenge for machines to process, curate, search and reason in such data.

The process of automatically identifying and disambiguating entities, their attributes and relationships in unstructured data sources is termed as *Information Extraction (IE)*. *IE* facilitates a rich and structured representation of data, enabling downstream applications to process unstructured documents in a manner similar to a standard database. The richness present in natural language text, presupposition of world knowledge, and the rapid rate of content creation makes IE a highly challenging task. As a result, it has been a very active area of research in the computational linguistics community since over two decades (Sarawagi, 2008).

1.1 Information Extraction : Challenges

Consider the snippets of news articles in Figure 1.1. The spans of text in boldface and italics are named-entities and relations mentioned in these text snippets, respectively. IE is a pre-processing step that annotates a given piece of text with the entities and relationships or creates a table – similar to a standard database – and populates it with the tuples of the facts extracted (shown on the right, in Figure 1.1). This facilitates the processing of unstructured data by text processing applications in a manner similar to that of standard database applications.

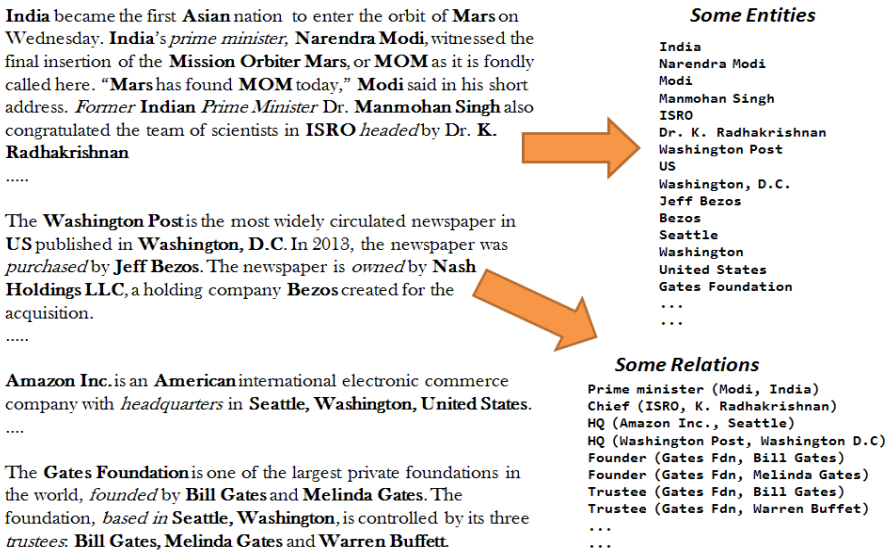


Figure 1.1: Text Snippets and Facts Extracted

A few of the challenges in IE are listed below (with illustrative examples wherever appropriate)

- Manual development and customization of rules, although shown to be very effective, are complex and labor-intensive tasks (Chiticariu et al., 2013).
- *Entity Disambiguation*: For example, `Jeff Bezos` and `Bezos` refer to the same entity. However, `Washington` could be a city, a state, or a person, depending on the context (Bunescu and Pasca, 2006; Kulkarni et al., 2009).
- *Scope resolution*: Certain entities such as `Washington` in "`Washington Post`" should not be labeled as a location name because the entire textual span is an organization name (Chiticariu et al., 2010b).
- *Type Disambiguation*: An entity, which is generally a location, in some contexts is an organization. For example, in the sentence, *"England beat Australia 2 – 0,"* `England` and `Australia` are sports organizations[1] (Chiticariu et al., 2010b).
- Creation of a large labeled training dataset for relation extraction is a costly affair. Existing hand-curated databases (for example, Freebase) can be leveraged (Mintz

[1]It can be argued that they are teams representing the countries. But the point is that there exists an ambiguity as to the type of the entity.

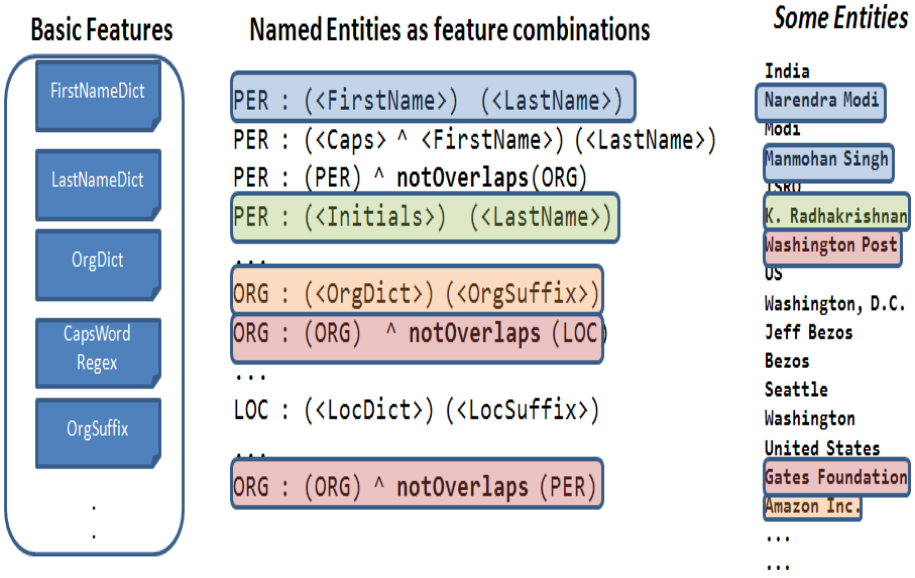


Figure 1.2: Patterns as Feature Combinations

et al., 2009; Riedel et al., 2010). However, this process presents a different set of challenges (a few of them mentioned below).

- *Temporal Validity*: Some database facts may not be currently valid. For example, `Prime Minister(Manmohan Singh, India)`[2].

- *Relation mention detection*: All sentences that contain an entity pair need not express the same relation that is present in the database. For instance, the co-occurrence of `Obama` and `US` in a sentence is not a sure indication that the `President` relation is expressed in it (Hoffmann et al., 2011; Surdeanu et al., 2012).

1.2 Overview of Thesis Contributions

The problem of IE can be viewed as that of finding patterns in data. These patterns can either take the form of features and/or can be specified as constraints on the search space.

[2]Some database facts are context dependent — both on time and type of relation. Temporal information extraction is an important area of research in the community (`http://l2r.cs.uiuc.edu/Talks/TIETutorial_Final.pdf`).

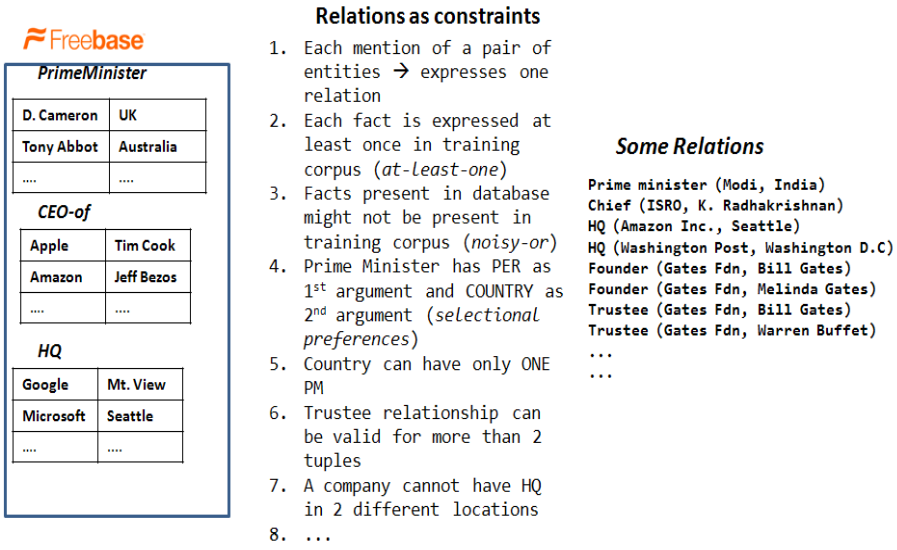


Figure 1.3: Patterns as Constraints

Data-driven Patterns : Feature Combinations

Let us suppose that we are given a set of basic features (for example, `Caps` - *a capitalized token*; `LastName` - *occurrence in a dictionary of last-names*). Named-entities can be discovered by learning combinations of such features. For instance, *"if a span of text contains two tokens,* `Caps` *followed by* `LastName`*, then it is most probably a person named-entity"*. We consider the previous statement as a pattern, leading to a named-entity.

Figure 1.2 depicts some of the basic features, a number of patterns (basic feature combinations) and the entities in text that can potentially match with these patterns. Named-entity recognition (NER) can immensely benefit from such patterns, some of which are domain-specific and others, domain-independent. Several patterns are non-trivial combinations of basic features. For instance, *"if a* `location` *name overlaps with an* `organization`*, then it is not a* `location` *named-entity"*. (for example, `Washington` in `Washington Post`). Such patterns are called as *relational features*.

In this thesis, we study the categorization of *relational feature* classes. We also define various methods to learn feature combinations through induction. The features induced are consumed by a rule-based NER system to learn compact and "interpretable" rules that have reasonable accuracy. We also demonstrate the use of these features in max-margin based sequence labeling models.

User-Specified Patterns : Constraints

Consider the problem of identifying relationships between entities in text. Here we can look at patterns as constraints that need to be enforced on the extracted relations. Some of these are listed in Figure 1.3. They are few compared to the entity-recognition case and can be specified by the user to restrict the search space.

For instance, we would like to enforce the following constraint: *For a* `Prime Minister` *relation, the first argument has to be a* `person` *and the second argument has to be a* `country`*.*

In this thesis, we look at a specific paradigm of relation extraction called *distant supervision* (Mintz et al., 2009). The goal is to learn relation extraction models by aligning facts in a database (Figure 1.3) to sentences in a large unlabeled corpus. Since the individual sentences are not hand labeled, the facts in the database act as "weak" or "distant" labels, and hence, the learning scenario is termed as "distantly supervised". We look at ways in which constraints can be specified while learning relation extractors in this setting. We formulate an integer linear programming-based framework to facilitate the addition of constraints.

Existing distant supervision-based systems are often trained by optimizing performance measures (such as conditional log-likelihood or error rate) that are not directly related to the task-specific non-linear performance measure, for example, the F_1-score. We present a novel max-margin learning approach to optimize non-linear performance measures for distantly supervised relation extraction models. Our approach can be more generally applied to a class of models that contain latent variables, quite prevalent in *natural language processing*.

1.3 Learning for Named-Entity Extraction

Several problems in Machine Learning are immensely benefited from a rich structural representation of the data (Flach and Lachiche, 1999; Roth and Yih, 2001). Specifically, the tasks in IE are relation-intensive and the usage of relational features has been shown to be quite effective in practice (Califf, 1998; Roth and Yih, 2001). In this section, we define categories of predicates and discuss the complexity-based classification of relational features followed by techniques to induce features in several of these categories.

Feature Space Categorization

The relational features are in a language that is similar to *first order definite clauses* in expressive power (Horn, 1951). Predicates are defined on textual spans. The combinations of predicates (clauses) can be viewed as rules of the form of *"if condition(s) then decision"*.

$$decision \quad :- \quad condition_1 \ \wedge \ condition_2 \ldots \ \wedge \ condition_n$$

The head predicate (*decision*) is the class label of a textual span. We define two types of body predicates ($condition_i$), namely, *relation* and *basic feature* predicates. A *relation* predicate is a binary predicate that represents the relationship between two *spans* of text. For example, `overlaps(X,Y)`. A *basic feature* predicate is an assertion of a situation or a property of a *span* or a *sub-span*. For example, `FirstName(`X`)` states that the span of text X occurs in a dictionary of first names. We illustrate each of these feature classes with an example of a typical definite clause that belongs to the feature class.

1. **Simple Conjuncts ($\mathcal{SC}$s)**: are simple conjunctions of basic features.
   `Org(X) :- OrgGazeteer(X),CapsWord(X).`

2. **Candidate Definition Features ($\mathcal{CD}$s)**: These consist of the following two feature classes.

   (a) **Absolute Features ($\mathcal{AF}$s)**: non-overlapping evidence predicates chained by relation predicates.
   `person-AF(X) :- contains(X, X1), FirstNameDict(X1),`
   `CapsWord(X1), before(X1,X2), contains(X, X2), CapsWord(X2).`

   (b) **Composite Features (CFs)**: Defined as a conjunction of two $\mathcal{AF}$s that share the same head predicate.
   `person(X) :- person-AF (X), leftContext(X, 1, L2),`
   `Salutation(L2).`

3. **Candidate Refinement Features ($\mathcal{CR}$s)**: The body of the clause is defined by head predicates that belong to different class labels, and can contain negations in the body (hence, not a definite clause)
   `Loc(X) :- Loc1(X),org1(Y),`$\neg$`overlaps(X,Y).`
   A span of text is a location, "*if it matches a location feature and does not overlap*

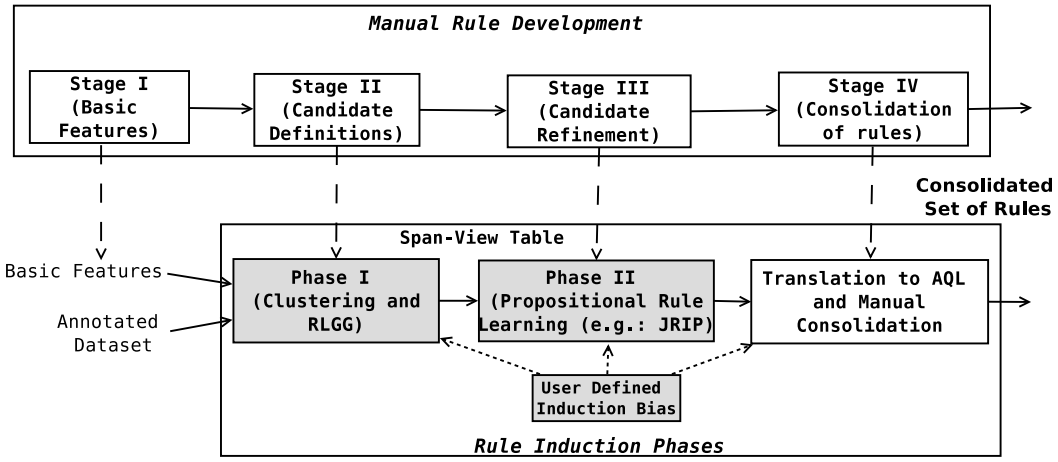


Figure 1.4: Phases of Rule Induction

with an organization feature". For example, due to this feature, Washington in "Washington Post" will not be marked as a location.

These feature classes are described in detail in Chapter 2.

1.3.1 Feature Induction in a Rule-based Setting

Rule-based systems for NER achieve state-of-the-art accuracies (Chiticariu et al., 2010b). However, manually building and customizing rules is a complex and labor-intensive process. In this work, we outline an approach that facilitates the process of building customizable rules for NER through rule induction. Given a set of basic feature predicates and an annotated document collection, our goal is to generate with reasonable accuracy an initial set of rules that are interpretable, and thus, can be easily refined by a human developer. Our contributions include (i) an efficient rule induction process in a declarative rule language, (ii) usage of induction biases to enhance rule interpretability, and (iii) definition of extractor complexity as a first step to quantify the interpretability of an extractor. We present initial promising results with our system and study the effect of induction bias and customization of basic features on the accuracy and complexity of induced rules. We demonstrate through experiments that the induced rules have good accuracy and low complexity according to our complexity measure.

Our induction system is modeled on a four-stage manual rule development process, since the overall structure of the induced rules must be similar in spirit to that which a developer who follows best practices would write. The stages of rule development and

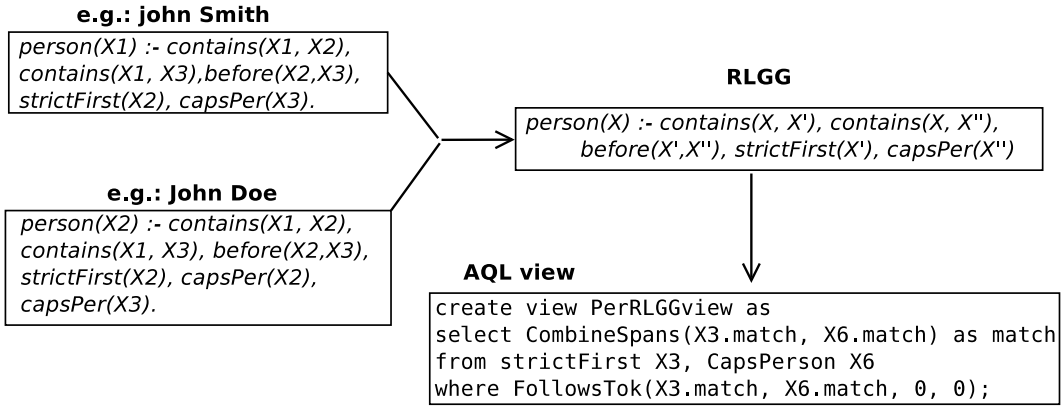


Figure 1.5: RLGG of an example pair

the corresponding phases of induction are summarized in Figure 1.4. In our system, we combine several induction techniques such as *relative least general generalization (RLGG)*, iterative clustering, and propositional rule learning in order to induce NER rules in a declarative rule language known as *Annotation Query Language (AQL)*.

A brief overview of the salient aspects of our induction system is presented in the following paragraphs.

Background Knowledge. We represent each example in the form of *first order definite clauses*, in conjunction with relevant background knowledge. This background knowledge will serve as the input to our induction system.

Clustering and RLGG. The first phase of induction uses a combination of *clustering* and *relative least general generalization (RLGG)* techniques (Nienhuys-Cheng and Wolf, 1997; Muggleton and Feng, 1990). Using clustering, we group the examples based on the similarity of their background knowledge. This process is interleaved by RLGG where we take a set of examples and find their generalization that is analogous to the *least upper bound*. We recursively find pairwise-RLGGs of all examples in a cluster. At the end of this phase, we have a number of $\mathcal{CD}$ features.

The representation of an example and the RLGG procedure is shown in Figure 1.5.

Propositional Rule Learning. In the second phase, we begin by forming a structure known as the *span-view table*. Broadly speaking, this is an attribute-value table formed by all the features induced in the first phase along with the textual spans generated by them. The attribute-value table is used as input to a propositional rule learner such as *JRIP* to learn accurate compositions of a useful subset (as determined by the learning

algorithm) of the $\mathcal{CD}$ features. This forms the second phase of our system. The rules learned from this phase are in the space of $\mathcal{CR}$ features.

Induction Biases. At various phases, several induction biases are introduced to enhance the interpretability of rules. These biases capture the expertise gleaned from manual rule development and constrain the search space in our induction system.

Extractor Complexity. Since our goal is to generate extractors with manageable complexity, we must introduce a quantitative measure of extractor complexity, in order to (1) judge the complexity of the extractors generated by our system, and (2) reduce the search space considered by the induction system. To this end, we define a simple complexity score that is a function of the number of rules, and the number of predicates in the body of each rule of the extractor.

AQL and SystemT : Advantages. The *hypothesis language* of our induction system is *AQL*, and we employ *SystemT* as the *theorem-prover*. SystemT provides a very fast rule execution engine and is crucial to our induction system because we test multiple hypotheses in the search for the more promising ones. AQL provides a very expressive rule representation language that has proven to be capable of encoding all the paradigms that any rule-based representation can encode (Chiticariu et al., 2011). The dual advantages of *rich rule-representation* and *execution efficiency* are the main motivations behind our choice.

We experimented with three different starting sets of basic feature predicates (with increasing accuracy and complexity) and observed that the complexity of the final set of induced rules is directly proportional to that of the initial set, both in terms of accuracy and complexity. We compared our induced set of rules with the manual rules. We achieve up to 75% accuracy of the state-of-the-art manual rules with a decrease in extractor complexity of up to 61%.

1.3.2 Feature Induction in a Max-margin Setting

In this piece of work, we view the problem of NER from the perspective of sequence labeling. The goal is to investigate the effectiveness of using relational features in the input space of a max-margin based sequence-labeling model. Our work is based on StructSVM Tsochantaridis et al. (2004) and StructHKL (Nair et al., 2012b) formulations. We propose two techniques to learn a richer sequence-labeling model by using the

relational features discussed above.

In one technique, we leverage the StructHKL system to learn optimal conjunctions (that is $\mathcal{SC}$s) of basic feature predicates. In another technique, we enumerate a good set of $\mathcal{CD}$s using existing induction techniques (such as RLGG and clustering) and use them as features in a StructSVM learning algorithm.

Our experiments in sequence labeling tasks reinforce the importance of induction bias and the need for interpretability to achieve high-quality NER rules, as observed in the experiments of our previous work on rule induction.

1.4 Learning for Relation Extraction

In the second part of the thesis, we investigate another important problem in IE, namely, *relation extraction*. The task of extracting relational facts that pertain to a set of entities from natural language text is termed as *relation extraction*. For example, given a natural language sentence, *"On Friday, President Barack Obama defended his administration's mass collection of telephone and Internet records in the United States"*, we can infer the relation, `President(Barack Obama, United States)` between the entities `Barack Obama` and `United States`.

Our framework is motivated by distant supervision for learning relation extraction models (Mintz et al., 2009). Prior work casts this problem as a multi-instance multi-label learning problem (Hoffmann et al., 2011; Surdeanu et al., 2012). It is multi-instance because for a given entity pair, only the label of the bag of sentences that contains both entities (or mentions) is given. It is multi-label because a bag of mentions can have multiple labels. The inter-dependencies between relation labels and (hidden) mention labels are modeled by a Markov Random Field (Figure 5.1) (Hoffmann et al., 2011).

Formally, the training data is $\mathcal{D} := \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ where $\mathbf{x}_i \in \mathcal{X}$ is the entity-pair, $\mathbf{y}_i \in \mathcal{Y}$ denotes the relation labels, and $\mathbf{h}_i \in \mathcal{H}$ denotes the *hidden* mention labels. The possible relation labels for the entity pair is observed from a given knowledge base. If there are L candidate relation labels in the knowledge base, then $\mathbf{y}_i \in \{0,1\}^L$, (for example, $y_{i,\ell}$ is 1 if the relation ℓ is licensed by the knowledge-base for the entity pair) and $\mathbf{h}_i \in \{1, .., L, nil\}^{|\mathbf{x}_i|}$ (that is, each mention realizes one of the relation labels or *nil*).

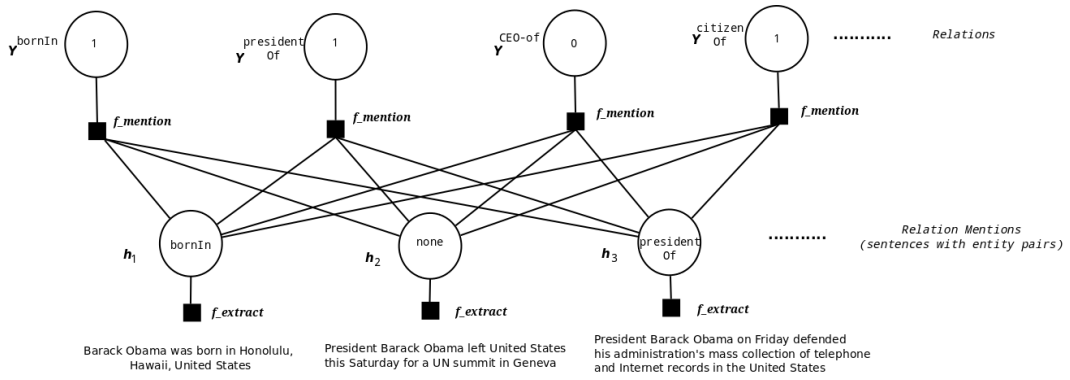


Figure 1.6: Graphical model for Distantly Supervised Relation Extraction.

1.4.1 Relaxed Distant Supervision

Various models have been proposed in recent literature to align the facts in the database to their mentions in the corpus. In this work, we discuss and critically analyze a popular alignment strategy called the *"at least one"* heuristic. We provide a simple, yet effective relaxation to this strategy.

Our work extends the work by Hoffmann et al. (2011). We formulate the inference procedures in training as integer linear programming (*ILP*) problems and implement the relaxation to the *"at least one"* heuristic through a soft constraint in this formulation. This relaxation is termed as *"noisy-or"*. The idea is to model the situation in which a fact is present in the database but is not instantiated in the text.

Empirically, we demonstrate that this simple strategy leads to a better performance under certain settings when compared to the existing approaches. Additionally, our inference formulation enables us to model additional types of constraints such as selectional preferences of arguments.

1.4.2 Distant Supervision in a Max-margin Setting

Rich models with latent variables are popular in many problems in natural language processing. For instance, in IE, one needs to predict the relation labels that an entity-pair can take based on the hidden relation mentions, that is, the relation labels for occurrences of the entity pair in a given corpus (as illustrated in Figure 5.1). These models are often trained by optimizing performance measures (such as conditional log-likelihood or error rate) that are not directly related to the task-specific non-linear performance measure, for example, the F_1-score. However, better models may be trained by optimizing the

task-specific performance measure, while allowing latent variables to adapt their values accordingly.

Large-margin methods have been shown to be a compelling approach to learn rich models detailing the inter-dependencies among the output variables. Some methods optimize loss functions decomposable over the *training instances* (Taskar et al., 2003; Tsochantaridis et al., 2004) compared to others that optimize non-decomposable loss functions (Ranjbar et al., 2013; Tarlow and Zemel, 2012; Rosenfeld et al., 2014; Keshet, 2014). They have also been shown to be powerful when applied to latent variable models when optimizing for decomposable loss functions (Wang and Mori, 2011; Felzenszwalb et al., 2010; Yu and Joachims, 2009).

In this work, we describe a novel max-margin learning approach to optimize non-linear performance measures for distantly supervised relation extraction models. Our approach can be generally used to learn latent variable models under multivariate non-linear performance measures, such as the F_β-score.

Our approach involves solving the hard-optimization problem in learning by interleaving the Concave-Convex Procedure with dual decomposition. Dual decomposition allowed us to solve the hard sub-problems independently. A key aspect of our approach involves a local-search algorithm that has led to a speed-up of 7,000 times in our experiments over an exhaustive search baseline proposed in previous work (Ranjbar et al., 2012; Joachims, 2005).

Our work is the first to make use of max-margin training in distant supervision of relation extraction models. We demonstrate the effectiveness of our proposed method compared to two strong baseline systems that optimize for the error rate and conditional likelihood, including a state-of-the-art system by Hoffmann et al. (2011). On several data conditions, we show that our method outperforms the baseline and results in an improvement of up to 8.5% in the F_1-score.

1.5 Thesis Organization

Our thesis can be summarized as shown in Figure 1.7. The broad theme of each work and the corresponding chapter/section in the thesis in which it is described, is indicated.

In the named-entity extraction setting, we work in the paradigm of *relational feature space exploration*. We describe and categorize relational features in Chapter 2. In Chap-

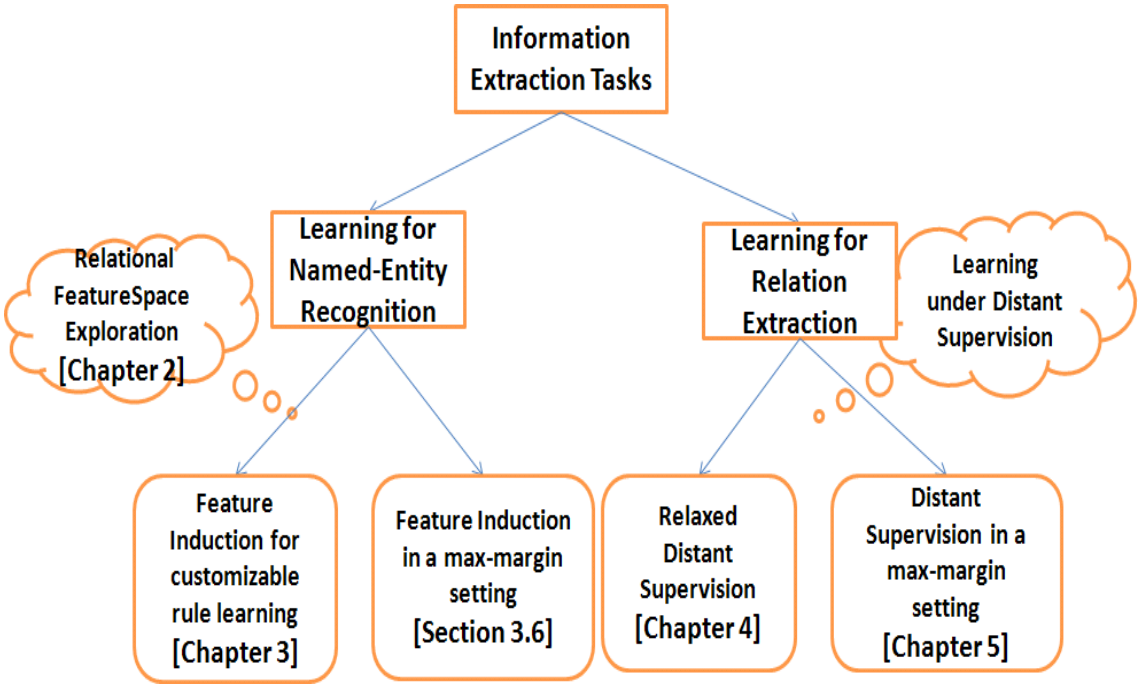


Figure 1.7: Thesis Organization

ter 3, we describe our approaches to learn NER rules in this feature space using induction techniques. Towards the end of this chapter in Section 3.6, we describe our experiments of employing relational features in max-margin based sequence-labeling models.

Subsequently, we switch gears a bit and describe another important IE task, namely, *relation extraction*. Here, our research has been in the paradigm of *learning under distant supervision*. In Chapter 4, we discuss a popular alignment strategy in distantly supervised relation extraction called the *"at least one"* heuristic. We provide a simple, yet effective relaxation to this strategy by the addition of constraints in the learning algorithm. Following this, in Chapter 5, we describe a novel max-margin learning approach to optimize non-linear performance measures for distantly supervised relation extraction models. We make our concluding remarks with pointers to directions of future research in Chapter 6.

Part I

Learning for Named Entity Recognition

Chapter 2

Relational Feature Space

Several problems in machine learning are immensely benefited from a rich structural representation of the data (Flach and Lachiche, 1999; Roth and Yih, 2001). Specifically, the tasks in Information Extraction are relation-intensive and the usage of relational features has been shown to be quite effective in practice (Califf, 1998; Roth and Yih, 2001).

As outlined in Chapter 1, relational features are in a language that is similar to *first order definite clauses* in expressive power (Horn, 1951). Predicates are defined on textual spans. The combinations of predicates (clauses) can be viewed as rules of the form of *"if condition(s) then decision"*.

$$decision \quad :- \quad condition_1 \ \wedge \ condition_2 \ldots \ \wedge \ condition_n$$

In this chapter, we define categories of predicates and discuss the complexity-based classification of relational features. In Appendix A, we make some claims and provide simple proofs of the hierarchy of relational features based on their complexity.

Some Definitions

We define a few important concepts that will be used in the rest of the thesis such as *span*, *predicate*, types of predicates, *definite clause*, and so on.

Definition 2.1. ***Span****: A span is an sequence of characters in natural language text. It has certain properties and is described by a set of attributes.*

For example, in Figure 2.1, *John Doe* is a span of text represented by X, and is a `PERSON` named-entity. It consists of sub-spans $X1$ and $X2$ each of which has a number of properties which are explained below.

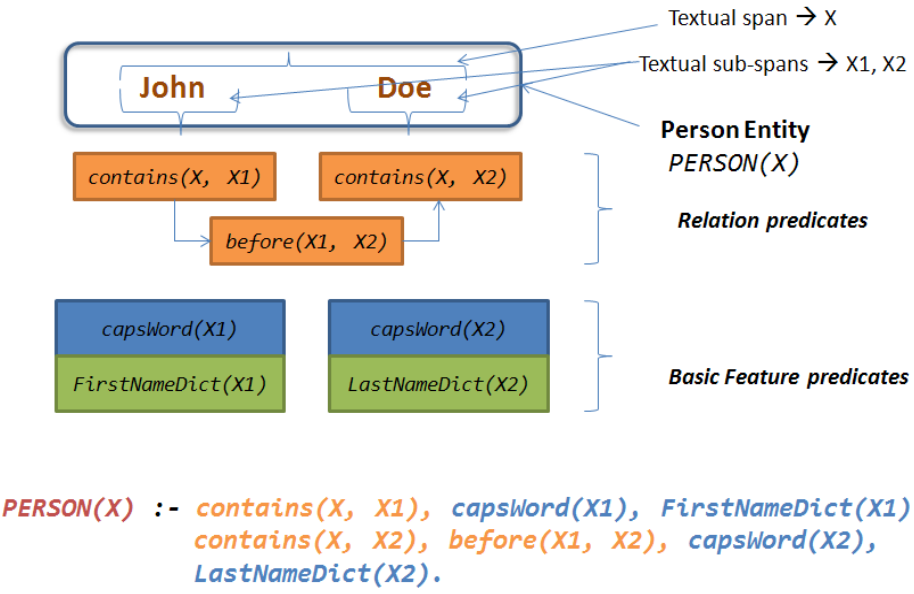


Figure 2.1: Types of Predicates

Definition 2.2. ***Predicate****: A predicate is a property of a span of text.*

For example, `OrgGazeteer(X)` is a predicate, which states that the span of text X is present in an `OrgGazeteer`[1]. A predicate gives rise to features used in our models. The predicates are composed to form clauses in *first-order logic*.

Definition 2.3. ***Clause****: A clause is an expression formed from a finite collection of predicates.*

An example is shown at the bottom of Figure 2.1. These clauses can be viewed as decision rules of the form *if condition then decision*. The decision part specifies the class label. The clause consists of a *head predicate* also called the *label predicate* (`PERSON(X)`) and a set of *body predicates* (composed as conjunctions) that describe the various properties of the span of text. Predicates that occur in the head are termed *positive* predicates and the one in the body are termed as *negative* predicates.

We define two types of *body predicates*, namely, *relation predicate* and *basic feature predicate*.

Definition 2.4. ***Relation Predicate****: A* relation *predicate is a binary predicate that represents the relationship between* spans *or between a* span *and its sub-span.*

[1] A collection of Organization names

Definition 2.5. ***Basic Feature Predicate****: A* basic feature *predicate is a unary predicate which denotes an assertion of a situation or a property of a* span *of text.*

The *relation* and *basic feature predicates* are similar to the *structural* and *property* predicates respectively in 1BC clauses (Flach and Lachiche, 1999).

The predicates take as arguments textual spans denoted by variables (e.g. X, $X1$). A *local variable* is one that occurs in the body of a clause (does not appear in the head predicate). In Figure 2.1, X represents the entire textual span. $X1$ is a *local variable* introduced by a relation predicate. A *local variable* is said to be *consumed* if it is used as an argument in any of the basic feature predicates. For instance, in Figure 2.1, $X1$ is *consumed* by `CapsWord` and `FirstNameDict` basic feature predicates.

Definition 2.6. ***Minimal Clause*** *: A minimal clause is one that cannot be constructed from smaller clauses that do not share common local variables.*

A clause with a single boolean predicate as the decision variable, is generally referred to as a *definite clause* (Horn, 1951).

Definition 2.7. ***Definite Clause****: A definite clause is one which consists of exactly one head predicate.*

Please refer to Figure 2.1 for an illustrated example of these definitions. We use the terms *features* and *predicates* interchangeably in the thesis.

2.1 Feature Classes

Most of the relational features that we describe in the thesis are in the space of definite clauses[2]. Based on the degree of expressive power, we categorize relational features as *simple conjuncts* ($\mathcal{SC}$), *candidate definition features* ($\mathcal{CD}$), and *candidate refinement features* ($\mathcal{CR}$). *Candidate definition features* ($\mathcal{CD}$) are in turn sub-categorized into *absolute features* ($\mathcal{AF}$), *primary features* ($\mathcal{PF}$), and *composite features* ($\mathcal{CF}$).

Simple Conjuncts ($\mathcal{SC}$):

$\mathcal{SC}$s are simple conjunctions of basic features (including unary conjunctions) of a single textual span. In other words, $\mathcal{SC}$s are conjunctions of only basic feature predicates. Figure 2.2 provides an illustrative example.

[2]Although *Candidate Refinement Feature* is not a definite clause

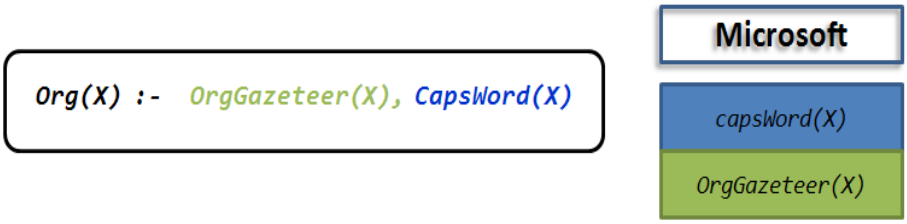


Figure 2.2: Simple Conjunct ($\mathcal{SC}$)

Candidate definition features ($\mathcal{CD}$)

Combinations of basic feature predicates along with relation predicates are termed as $\mathcal{CD}$s. They encodes sequence information that is quite prevalent in IE tasks. $\mathcal{CD}$s are turn sub-categorized into:

1. Absolute $\mathcal{CD}$ Features ($\mathcal{AF}$)
2. Primary $\mathcal{CD}$ Features ($\mathcal{PF}$)
3. Composite $\mathcal{CD}$ Features ($\mathcal{CF}$)

Each of these is elaborated below.

Absolute $\mathcal{CD}$ Features ($\mathcal{AF}$):

In an absolute feature, new local variables can be introduced only in a *relation* predicate. Any number of *relation* and *basic feature* predicates can be conjoined to form an $\mathcal{AF}$, such that, the resultant $\mathcal{AF}$ is minimal and the local variables introduced in the *relation* predicates are (transitively) *consumed* by some *basic feature* predicates.

Unlike the 1BC clauses, any number of new local variables can be introduced in a *relation* predicate. Figure 2.3 depicts an $\mathcal{AF}$. Note that this feature class does not contain contextual clues, in contrast to $\mathcal{CF}$ (defined below).

Primary $\mathcal{CD}$ Features ($\mathcal{PF}$):

Primary features are $\mathcal{AF}$s in which a new local variable introduced is consumed only once[3][4]. In other words, in a $\mathcal{PF}$, every relation predicate has only one basic feature predicate. An illustration is provided in Figure 2.4.

[3] This is similar to elementary features in (Flach and Lachiche, 1999) except that elementary features allow only one new local variable in a *structural* predicate.

[4] $\mathcal{PF}$s are different from simple clauses (McCreath and Sharma, 1998), as a simple clause allows a local variable to be unconsumed.

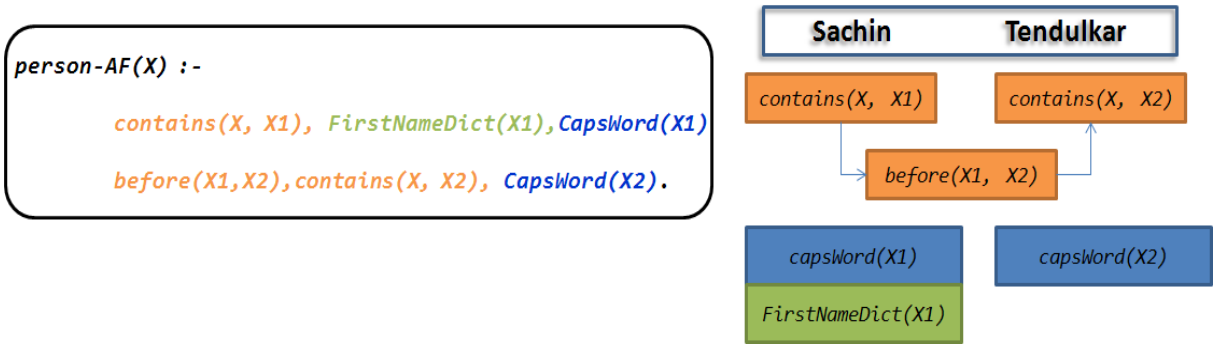


Figure 2.3: Absolute Feature ($\mathcal{AF}$)

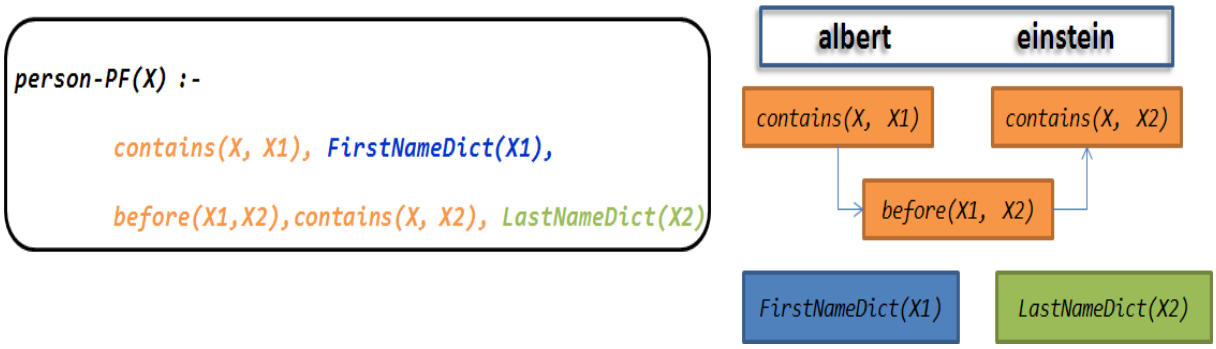


Figure 2.4: Primary Feature ($\mathcal{PF}$)

Composite $\mathcal{CD}$ Features ($\mathcal{CF}$):

Composite Features are features formed by the conjunction of one or more $\mathcal{AF}$s without anti-unification [5] of body literals. Only the head predicates are unified. As in $\mathcal{AF}$s, every local variable introduced in a relational predicate should be consumed by other relation or basic feature predicates.

Figure 2.5 illustrates a $\mathcal{CF}$. Note that a $\mathcal{CF}$ includes additional contextual clues like `leftContext` and `Salutation` which are predicates over a span of text that is not a part of the head predicate variable.

Candidate Refinement Features ($\mathcal{CR}$s):

Candidate refinement features are defined using $\mathcal{CD}$s that contain different label predicates. They are used to discard spans output by the $\mathcal{CD}$ features that may be incorrect. Hence it is termed as refinement. An illustration of a $\mathcal{CR}$ with an example clause is

[5] Anti-unification is the process of constructing a generalization common to two given symbolic expressions. For more details please refer to https://en.wikipedia.org/wiki/Anti-unification_(computer_science)

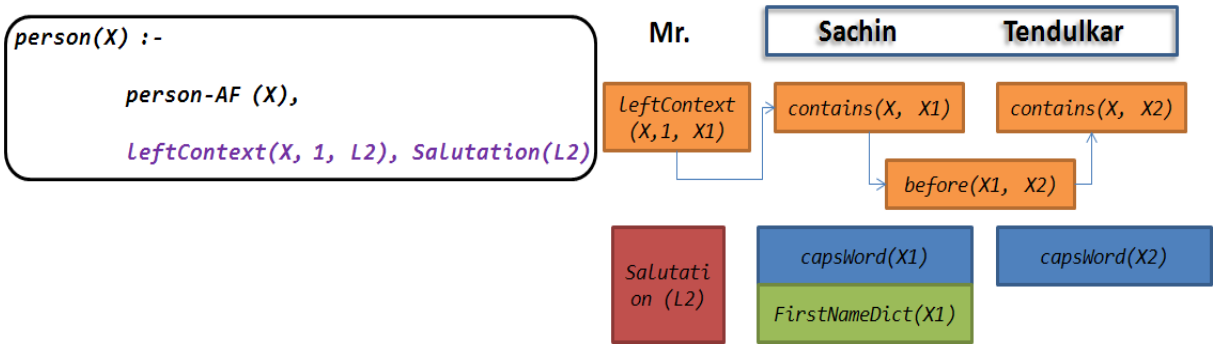


Figure 2.5: Composite Feature ($\mathcal{CF}$)

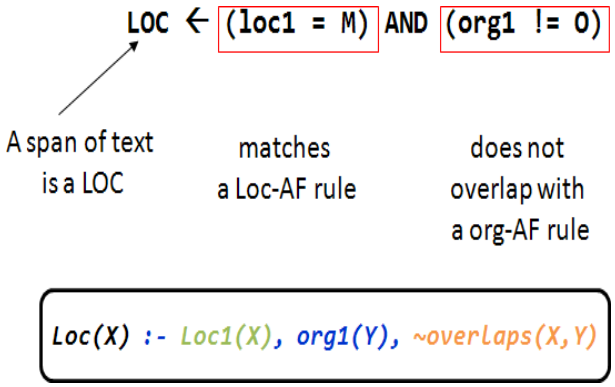


Figure 2.6: Candidate Refinement Feature ($\mathcal{CR}$)

provided in Figure 2.6. It is read as follows: A span of text is a location, "*if it matches a location feature and does not overlap with an organization feature*". For example, due to this feature, Washington in "Washington Post" will not be marked as a location.

A $\mathcal{CR}$ is not a definite clause since it contains more than one predicate which is positive as seen from the example.

To summarize, we use the following set of definite clause examples to illustrate the various feature classes, for easy reference.

1. `ORG(X) :- OrgGazeteer(X).`
2. `ORG(X) :- OrgGazeteer(X), CapsWord(X).`
3. `ORG(X) :- contains(X, X1), CapsWord(X1).`
4. `ORG(X) :- contains(X, X1).`
5. `ORG(X) :- contains(X, X1), CapsWord(X1). FirstNameDict(X1).`

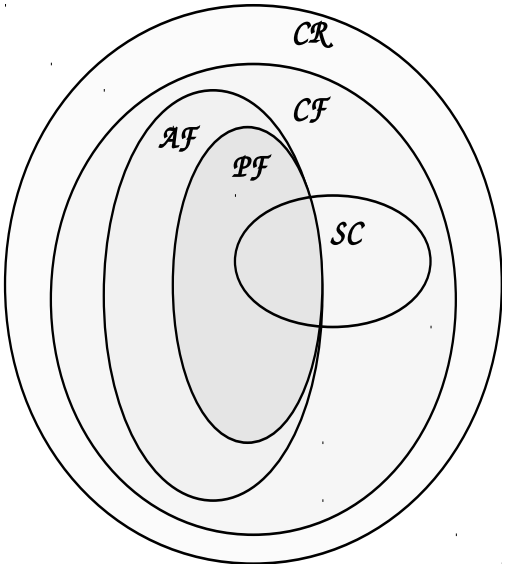


Figure 2.7: Hierarchy of Features

$$\mathcal{PF} \subset \mathcal{AF} \subset \mathcal{CF}$$

$$\mathcal{SC} \subset \mathcal{CF}$$

$$\mathcal{SC} + \mathcal{PF} + \mathcal{AF} + \mathcal{CF} = \mathcal{CD}$$

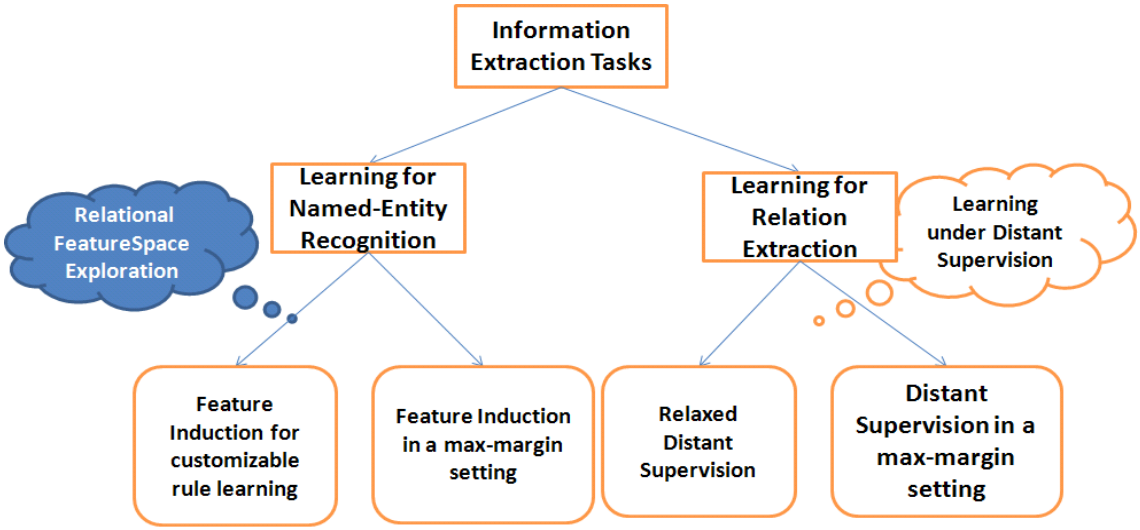
6. `PER(X) :- contains(X, X1), CapsWord(X1), contains(X,X2), before(X1, X2), LastNameDict(X2).`

7. `PER(X) :- contains(X, X1), FirstNameDict(X1), CapsWord(X1), before(X1,X2), contains(X, X2), CapsWord(X2), leftContext(X, 1, L2), Salutation(L2).`

Clauses 1 and 2 above are $\mathcal{SC}$s. Clauses 1, 3, 5 and 6 above are $\mathcal{AF}$s, whereas, clauses 2 (not minimal), 7 (not minimal) and 4 (since variable `X1` is not consumed) are not. Clauses 1, 2, 3, 5, 6 and 7 are $\mathcal{CF}$s, whereas, clause 4 is not. Only clauses 1 and 3 are $\mathcal{PF}$s.

The feature classes defined here form a hierarchy as shown in Figure 2.7. The relationship among various feature classes are provided in Appendix A.

Figure 2.8 highlights the part presented in this chapter and the accompanying publications that overlap with the material presented. In the next chapter, we discuss our techniques to induce relational features for the named-entity recognition task.



Naveen Nair, Ajay Nagesh and Ganesh Ramakrishnan. **Probing the Space of Optimal Markov Logic Networks for Sequence Labeling**. In 22nd International Conference on Inductive Logic Programming (**ILP**), Dubrovnik, 2012

Ajay Nagesh, Naveen Nair and Ganesh Ramakrishnan. **Comparison between Explicit Learning and Implicit Modeling of Relational Features in Structured Output Spaces**. In 23rd International Conference on Inductive Logic Programming (**ILP**),Rio de Janeiro, Brazil, 2013.

Figure 2.8: Thesis Organization: Chapter 2

Chapter 3

Relational Feature Induction in a Rule-based Setting

In the previous chapter, we defined relational features and categorized them into various classes. In this chapter, we discuss the use of such features for the identification of named-entities in text. Our focus for the most part of this chapter will be a rule-based setting for entity extraction. We begin describing the rule-based approach for entity recognition and highlight its importance and challenges involved.After a brief discussion on rule-based entity recognition system, SystemT (and its rule language (AQL), we define our notion of rule complexity in 3.3. We then present our approach for inducing $\mathcal{CD}$ and $\mathcal{CR}$ rules, and discuss induction biases that would favor interpretability (Section 3.4) and discuss the results of an empirical evaluation (Section 3.5). Subsequently, we discuss some of the shortcomings of our complexity score in fully capturing extractor interpretability. This is followed by our experiments of using the relational features in max-margin based sequence-labeling models. In Appendix B we provide more details on SystemT and AQL. We also define the target language for our induction algorithm in the AQL language in Appendix C.

3.1 Entity Recognition : A Rule-based Perspective

Named-entity recognition (NER) is the task of identifying and classifying mentions of entities occurring in text with one or more rigid designators. Rigid designators include proper names as well as certain natural kind terms like biological species and substances (Nadeau and Sekine, 2007; Kripke, 1980). For example, identifying proper nouns such as names of persons, organizations, locations, products, brands, chemicals, proteins, and so on fall under the purview of the NER task.

System	Dataset	$\mathbf{F}_{\beta=1}$	
		Generic	**Customized**
GATE	ACE2002	57.8	82.2
SystemT	ACE 2005	57.32	88.95
	CoNLL 2003	64.15	91.77
	Enron	76.53	85.29

Table 3.1: Accuracies of rule-based NER systems

NER is a very challenging task because it presupposes world knowledge (in some cases, deep domain knowledge) to determine whether a given span of text is indeed a named-entity and to identify its type. Since the body of human knowledge is increasing at a rapid pace, new terminologies are constantly added to our lexicon. This necessitates the design of systems that can automatically identify such entities in text. NER systems discussed in literature roughly fall under three categories :

1. Rule-based systems. (Krupka and Hausman, 1998; Sekine and Nobata, 2004; Patel et al., 2010; Riloff, 1993; Soderland, 1999; Califf and Mooney, 1999, 1997; Reiss et al., 2008; Chiticariu et al., 2010b)

2. Statistical or Machine Learning-based systems. (Bender et al., 2003; Florian et al., 2003; McCallum and Li, 2003; Finkel and Manning, 2009; Singh et al., 2010)

3. Hybrid solutions (Srihari et al., 2000; Jansche and Abney, 2002)

Generic NER rules have been shown to work reasonably well out-of-the-box, and with further domain customization (Chiticariu et al., 2010b), achieve quality that surpasses state-of-the-art results obtained with statistical approaches.

Table 3.1 summarizes the quality of NER rules out-of-the-box and after domain customization in the GATE (Cunningham et al., 2011) and SystemT (Chiticariu et al., 2010a) systems, as reported in Maynard et al. (2003) and Chiticariu et al. (2010b), respectively, on several domains.

While statistical approaches have their own advantages, rule-based systems are still widely used in enterprise settings for two main reasons: *explainability* and *customizability*. Rules are transparent, which leads to better explainability of errors. One can easily identify the cause of a false positive or negative, and refine the rules without affecting other correct results identified by the system. Furthermore, an IE developer can understand rules more easily; they can also be customized for a new domain without requiring additional labeled data.

3.1.1 Building Blocks of NER rules

Typically, a rule-based NER system consists of a combination of four categories of rules (Chiticariu et al., 2010b). Along with the description of these categories, we also map these rule categories to the feature class hierarchy discussed in Chapter 2, (when appropriate).

1. *Basic Features (BF)* rules to identify components of an entity, such as, first name and last name. These correspond to the basic feature predicates in the feature class hierarchy.

2. *Candidate definition ($\mathcal{CD}$)* rules to identify complete occurrences of an entity by combining the output of multiple BF rules, for example, *first name followed by last name is a person candidate*. These correspond to the Candidate Definition features. They encompass all types of Candidate Definition features, namely, Primary $\mathcal{CD}$ features, Absolute $\mathcal{CD}$ features (without context), and Composite $\mathcal{CD}$ features (with context).

3. *Candidate refinement ($\mathcal{CR}$)* rules to refine candidates generated by $\mathcal{CD}$ rules. For example, *discard candidate persons contained within organizations*. These correspond to refinement features in the feature hierarchy.

4. *Consolidation rules (CO)* to resolve overlapping candidates generated by multiple $\mathcal{CD}$ and $\mathcal{CR}$ rules.

3.1.2 Annotator Development Lifecycle

Developing named-entity recognition rules with high accuracy and coverage is an iterative and labor-intensive process. This process is broadly divided into three phases: *develop-*

ment, testing and *analysis* (Chiticariu et al., 2011). We call this *annotator development lifecycle*, and depict this process in Figure 3.1. These phases involve the following six stages:

1. Definition of basic features such as dictionaries and regular expressions.
2. Development of an initial set of rules that combine these basic features to generate candidates rules.
3. Development of filtering rules and other candidate refinement rules.
4. Consolidation of rules, within a single type, and across types.
5. Examination of the output of the rules: error identification (false positives and false negatives), and analysis.
6. Iterative refinement of the features and rules developed in previous steps until the desired quality of the extractor is reached.

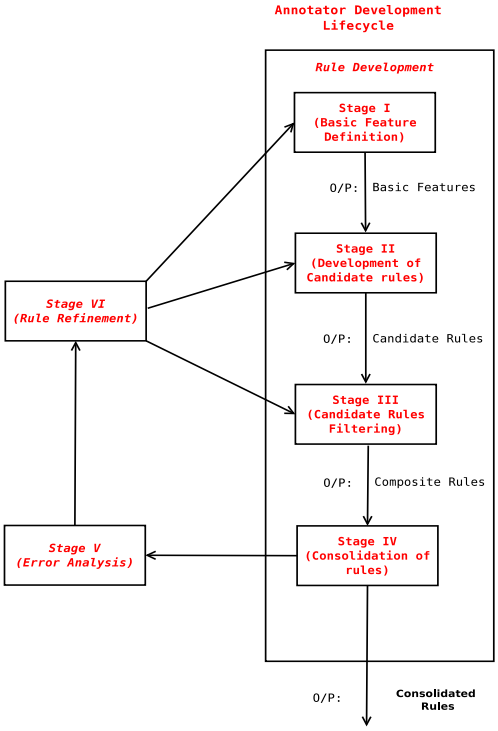


Figure 3.1: Annotator Development Lifecycle

The process is highly iterative as the requirements of the annotator keeps changing frequently.The involvement of the developers is crucial in the entire process. A detailed explanation of the stages mentioned above including the challenges involved in them is presented in Chiticariu et al. (2010b) and Liu et al. (2010).

Automating all or at-least some of these stages is extremely desirable. However, it is imperative that this be accomplished with a human in the loop, for the following reasons: (i) to avoid overfitting, (ii) to keep the rules simple and easy to maintain, and (iii) to produce rules that generalize well outside the corpus when appropriate (for example, *go from a few examples of sports-team names, to a complete dictionary of sports teams*).

A well-known drawback that influences the adoptability of rule-based NER systems is the manual effort required to build the rules. A common approach to address this problem is to build a generic NER extractor, and, then customize it for specific domains. While this approach partially alleviates the problem, substantial manual effort (in the order of several person weeks) is still required for the two stages, as reported in Maynard et al. (2003) and Chiticariu et al. (2010b). In this Chapter, we present our research work towards facilitating the process of building a generic NER extractor using induction techniques.

3.1.3 Problem Statement and Research Contributions

The input to our system is an annotated document corpus (annotated with named entities – PER, ORG and LOC), a set of basic feature predicates and a default CO rule for each entity type. Our goal is to induce a set of $\mathcal{CD}$ and $\mathcal{CR}$ features, such that the resulting extractor constitutes a good starting point for further refinement by a developer. Since the generic NER extractor has to be manually customized, a major challenge is to ensure that the induced rules have good accuracy, and at the same time, that they are *not too complex*, and consequently *interpretable*. Intuitively, an extractor that consists of too many rules, or has rules that are very complex or have complex interdependencies is difficult to understand, refine or customize. The main contributions in this work are

1. An *efficient* system for NER rule induction, using a highly expressive rule language (AQL) as the target language. The first phase of rule induction uses a combination of *clustering* and *relative least general generalization (RLGG)* techniques to learn $\mathcal{CD}$ features. The second phase identifies $\mathcal{CR}$ features using a propositional rule learner such as *JRIP* to learn accurate compositions of $\mathcal{CD}$ features.

2. The use of induction biases to enhance the interpretability of the induced rules. These biases capture the expertise gleaned from manual rule-development and con-

strain the search space in our induction system.

3. Definition of an initial notion of extractor complexity to quantify the interpretability of an extractor, and to guide the process of adding induction biases in order to favor learning *less complex* extractors. This is to ensure that the rules are easily customizable by the developer.

3.1.4 Comparison to Related Work in Induction

Existing approaches to rule induction for Information Extraction focus on rule-based systems based on the cascading grammar formalism exemplified by the Common Pattern Specification Language (CPSL) (Appelt and Onyshkevych, 1998), in which rules are specified as a sequence of basic features that describe an entity, with limited predicates in the context of an entity mention. Patel et al. (2010) and Soderland (1999) elaborate on top-down techniques for induction of IE rules, whereas, Califf and Mooney (1997, 1999) discuss a bottom-up IE rule induction system that uses the *relative least general generalization (RLGG)* of examples[1]. However, in all these systems, the expressivity of the rule-representation language is restricted to capturing sequence information. Contextual clues and higher-level rule interactions such as *filtering* and *join* are very difficult, if not impossible, to express in such representations without resorting to custom code [2]. Learning higher-level interactions between rules has received little attention. Our technique for learning higher-level interactions is similar to the induction of *ripple down rules* (Gaines and Compton, 1995), which, to the best of our knowledge, has not been previously applied to IE. A framework for refining AQL extractors based on an annotated document corpus has been described in Liu et al. (2010). We present complementary techniques to induce an initial extractor that can be automatically refined in this framework.

3.2 SystemT and AQL

SystemT is a declarative Information Extraction system based on an algebraic framework. In SystemT, developers write rules in an *SQL*-like language called *Annotation Query*

[1]Our work also makes use of RLGGs but computes these generalizations for clusters of examples, instead of pairs.

[2]Please refer to Section B of the Appendix for more details.

Language (AQL). To represent annotations in a document, AQL uses a simple relational data model with three types, namely:

1. *Span*: A region of text within a document identified by its "begin" and "end" positions.
2. *Tuple*: A fixed-size list of spans.
3. *Relation*, or *View*: A multi-set of tuples, where every tuple in the view must be of the same size.

A simple example of an AQL statement is given below. It is an extract statement[3] defined on the `Document` view. It uses the `regex` AQL construct to extract simple candidate person name tokens from `Document`.

```
create view CapsPerson as
extract
    regex /[A-Z](a-zA-Z|-)*/
on D.text as caps
from Document D;
```

Internally, SystemT compiles an AQL extractor into an executable plan in the form of a graph of *operators*. The formal definition of these operators takes the form of an algebra (Reiss et al., 2008), that is similar to relational algebra, but with extensions for text processing. The decoupling between AQL and the operator algebra allows for greater rule expressivity because the rule language is not constrained by the need to compile to a finite state transducer, such as in grammar systems based on the CPSL standard. In fact, join predicates such as *Overlaps*, as well as filter operations (*minus*) are extremely difficult to express in CPSL systems such as GATE without an escape to custom code (Chiticariu et al., 2010b). In addition, the decoupling between the AQL specification of *"what"* to extract from *"how"* to extract it, allows greater flexibility in choosing an efficient execution strategy among the many possible operator graphs that may exist for the same AQL extractor. Therefore, extractors written in AQL achieve a throughput that is orders of magnitude higher than traditional rule-based systems (Chiticariu et al., 2010a).

[3]The main types of statements in AQL are briefly described in Appendix B

3.3 Extractor Complexity

Since our goal is to generate extractors with manageable complexity, we must introduce a quantitative measure of extractor complexity, in order to:

- Judge the complexity of the extractors generated by our system; and
- Reduce the search space considered by the induction system

To this end, we define a simple complexity score that is a function of the number of rules, and the number of predicates in the body of a rule. In particular for AQL constructs the complexity is the number of input views to each rule of the extractor[4]. We define the *length of rule* R, denoted as $L(R)$, as the number of input views in the *from* clause(s) of the view. For example, in Figure B.1, we have $L(R_4) = 2$ and $L(R_5) = 3$, since R_4 and R_5 have two, and respectively, three views in the *from* clause. Furthermore, $L(R_8) = 2$ since each of the two inner statements of R_8 has one *from* clause with a single input view. The complexity of BF rules (for example, R_1 to R_3) and CO rules (for example, R_9) is always 1, since these types of rules have a single input view. We define the *complexity of extractor* E, denoted as $C(E)$ as the sum of lengths of all rules of E. For example, the complexity of the *Person* extractor from Figure B.1 is 15 plus the length of all rules involved in defining *Organization*, which are omitted from the figure.

Our simple notion of rule length is motivated by existing literature in the area of database systems (Abiteboul et al., 1995), where the *size* of a conjunctive query is determined only by the number of atoms in the body of the query (for example, items in the FROM clause); it is independent on the number of join variables (that is, items in the WHERE clause), or the size of the head of the query (for example, items in the SELECT clause). As such, our notion of complexity is rather coarse, and we shall discuss its shortcomings in detail in Section 3.5.3. However, we shall show that the complexity score significantly reduces the search space of our induction techniques, leading to simpler and smaller extractors, and therefore constitutes a good basis for more comprehensive measures of interpretability in the future.

[4]Since the target language of rule induction is AQL, we have defined complexity using AQL constructs. Please refer to Figure B.1 in Appendix B for the details.

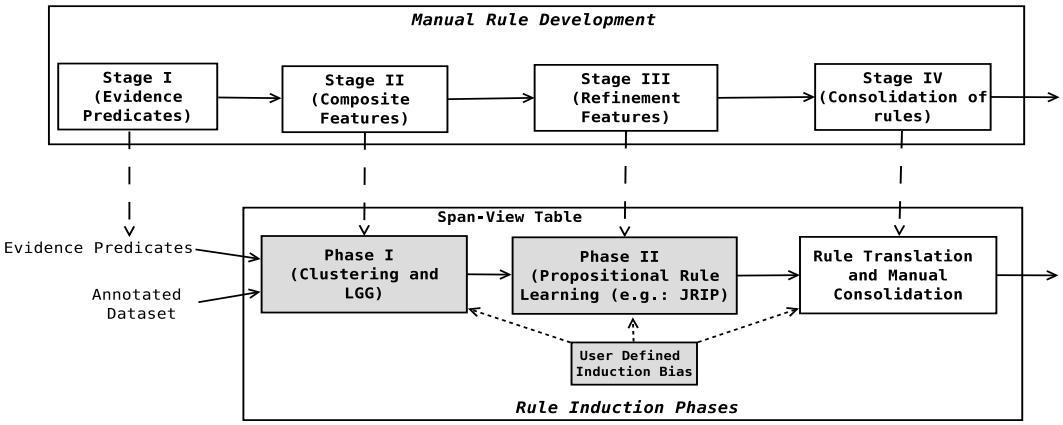


Figure 3.2: Correspondence between Manual Rule development and Rule Induction.

3.4 Methodology : Process of Rule Induction

Since the goal is to generate rules that can be customized by humans, the overall structure of the induced rules must be similar in spirit to that which a developer who follows best practices would write. Hence, the induction process is divided into multiple phases. Figure 3.2 shows the correspondence between the phases of induction and the types of rules. In Table C.1 (Appendix C), we summarize the phases of our induction algorithm, along with the subset of AQL constructs that comprise the language of the rules learned in that phase, the possible methods prescribed for inducing the rules and their correspondence with the stages in the manual rule development.

Our induction system generates rules for two of the four categories, namely $\mathcal{CD}$ and $\mathcal{CR}$ rules, as highlighted in Figure 3.2. We assume that we are given the BFs in the form of dictionaries and regular expressions. Prior work on learning dictionaries (Riloff, 1993) and regular expressions (Li et al., 2008) could be leveraged to semi-automate the process of defining the basic features.

The overall architecture of the system is shown in Figure 3.3. Next, we discuss our induction procedure in detail.

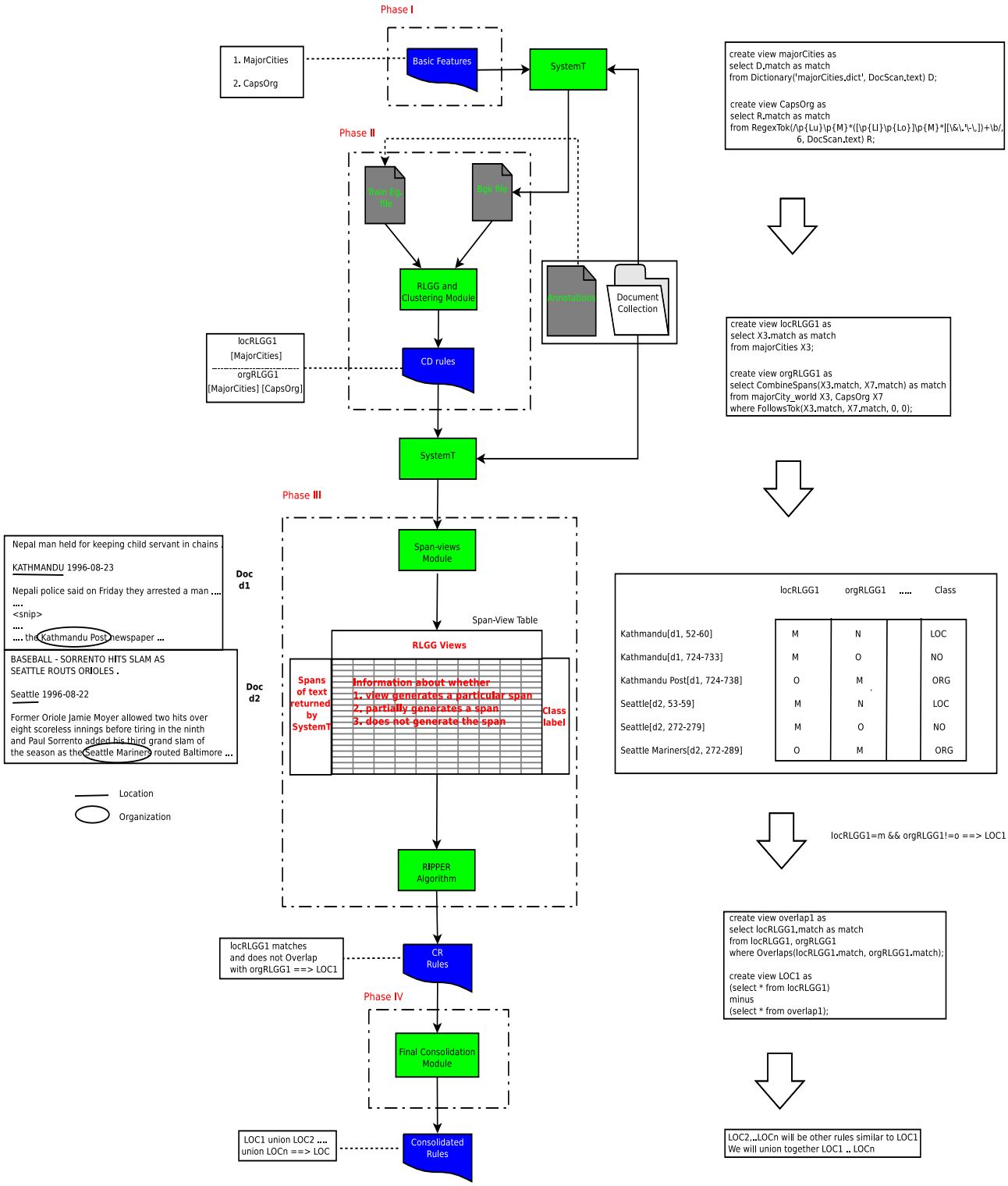


Figure 3.3: System Diagram with details and illustrative examples.

3.4.1 Basic Features and Background Knowledge

We assume that we are provided with a set of dictionaries and regular expressions to define all our basic features (basic feature predicates defined in Chapter 2)[5] These basic

[5]They are defined using *create view* statements in AQL. R_1, R_2 and R_3 in Figure B.1 of appendix are such basic view definitions.

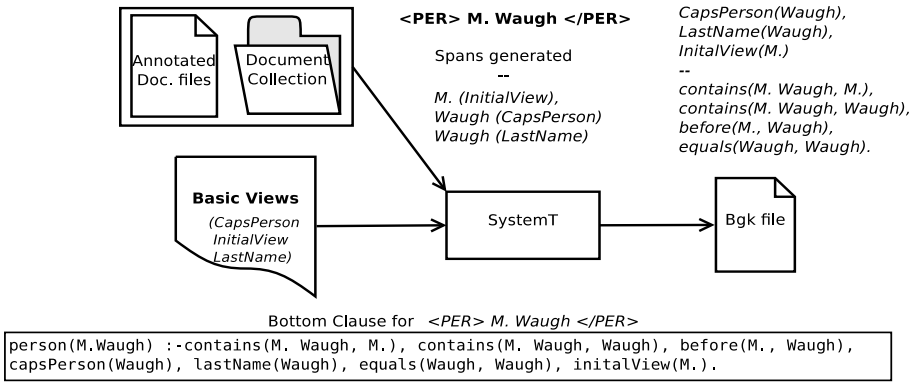


Figure 3.4: Background Knowledge Creation

views are compiled and executed in SystemT over the training document collection, and the resulting spans are represented by equivalent predicates in first order logic.

Essentially, each training example is represented as a definite clause that includes in its body the basic view-types encoded as background knowledge predicates. To establish relationships between different background knowledge predicates for each training example, we define *relation predicates*, discussed in Chapter 2. Some examples of such predicates are `contains`, `equals` and `before`.

This process is illustrated with an example in Figure 3.4. The training instance is `<PER> M. Waugh </PER>`, and the basic views return the spans *M.* (`initialView`), *Waugh* (`capsPerson`) and *Waugh* (`lastName`). The resultant background knowledge predicates is shown in the textbox at the bottom of Figure 3.4.

We observed there is some semantic overlap between the basic features of different types (PER, ORG, LOC). This is especially true for regular expression-based features. As a result of this, there might be some generalizations that contain non-intuitive compositions of basic features (For example, *a person generalization will contain an organization regular expression.*) To enhance rule interpretability, we partition the set of dictionaries and regular expressions across the entity types. This way, all $\mathcal{CD}$ rules for the *person* annotator will invoke only dictionaries and regular expressions that have been specifically set aside for induction for the *person* type, and likewise for each of the other types.

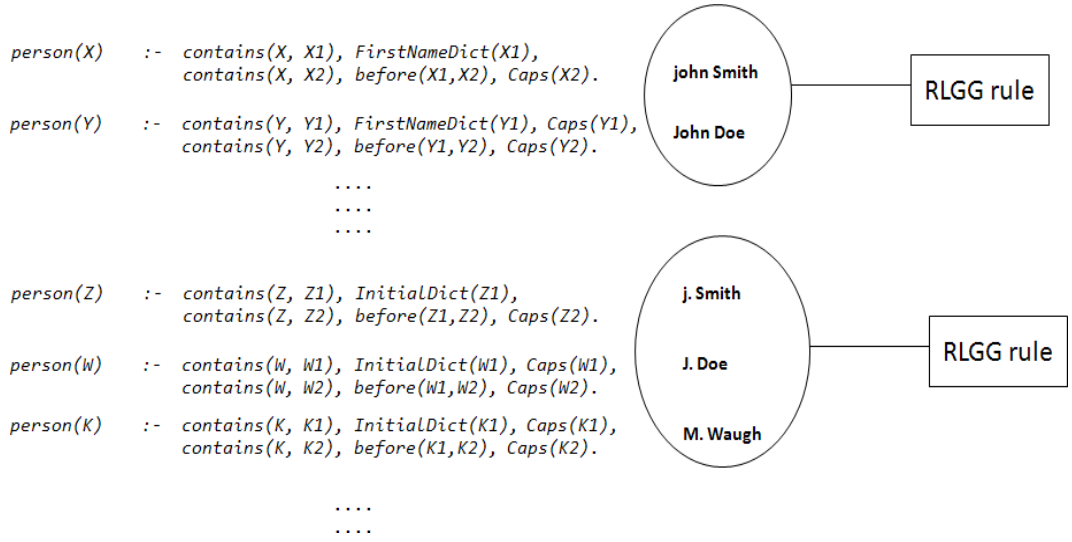


Figure 3.5: Clustering of Examples

3.4.2 Induction of Candidate Definition Features

Clustering Module

Named entities of a particular type can be categorized into a set of paradigms by virtue of their surface patterns in natural language text. For instance, in the case of *person* entities, a few of such patterns are listed below:

- `<FirstName> <LastName>`. *e.g. Mark Waugh ; John Doe*
- `<Initials> <LastName>`. *e.g. M. Waugh ; J. Doe*
- `<LastName> ,<FirstName>`. *e.g. Waugh, Mark ; Doe, John*
- `<FirstName> <MiddleName> <LastName>`. *e.g. Mark Anderson Waugh*

It is worthwhile to look at generalizations of instances that are similar. For instance, two-token person names such as *Mark Waugh* and *Mark Twain* have a distinct set of predicates compared to names that have initials in them as shown in Figure 3.5. However, we would not be able to generalize a two-token name (for example, *Mark Waugh*) with another name consisting of initials followed by a token (for example, *M. Waugh*). Such pairing of examples would not generate any meaningful generalization, resulting in a computational overhead. Given N examples, there would be $O(N^2)$ pairs that would have to be considered for generalization (Muggleton and Feng, 1990). We avoid this overhead by clustering examples as shown, in Figure 3.5.

We obtain non-overlapping clusters of examples within each type, by computing similarities between their representations as definite clauses. The clustering module is implemented as a wrapper around the hierarchical agglomerative clustering in LingPipe[6]. We use a custom similarity function that uses the following information:

- the number of common predicates; and
- the association between different predicates using global co-occurrence.

Specifically, we define the similarity function between two examples e_k and $e_{k'}$ consisting of a set of features $\left\{e_k^{BF_1}, e_k^{BF_1}, e_k^{BF_2}, \ldots, e_k^{BF_l}\right\}$ and $\left\{e_{k'}^{BF_1}, e_{k'}^{BF_2}, \ldots, e_{k'}^{BF_m}\right\}$ respectively, as follows:

$$S = \sum_{i,j} score\left(e_k^{BF_i}, e_{k'}^{BF_j}\right)$$

where $score\left(e_k^{BF_i}, e_{k'}^{BF_j}\right)$ is defined as the normalized frequency of co-co-occurrence of features BF_i and BF_j in the training dataset. In other words, the similarity function is sum of scores of all feature pairs in the examples considered. Intuitively, this function brings together examples that have similar set of features occurring in them and therefore examples which are similar to each other in each cluster.

As a consequence, we look at generalizations only within each cluster. So, clustering improves the efficiency by reducing the computational overhead apart from providing meaningful generalizations.

RLGG computation

We compute our $\mathcal{CD}$ features as the *relative least general generalization (RLGG)* (Nienhuys-Cheng and Wolf, 1997; Muggleton and Feng, 1990) of examples in each cluster. Given a set of clauses in first order logic, their RLGG is defined as follows:

Definition 3.1. *RLGG: The least generalized clause in the subsumption lattice of the clauses relative to the background knowledge (Nienhuys-Cheng and Wolf, 1997).*

It can be proved that in a subsumption lattice over definite clauses, the relative LGG of two examples with respect to a background knowledge, comprises only ground literals (as in our case), is the same as the LGG of the corresponding two bottom clauses (Plotkin,

[6] `http://alias-i.com/lingpipe/demos/tutorial/cluster/read-me.html`

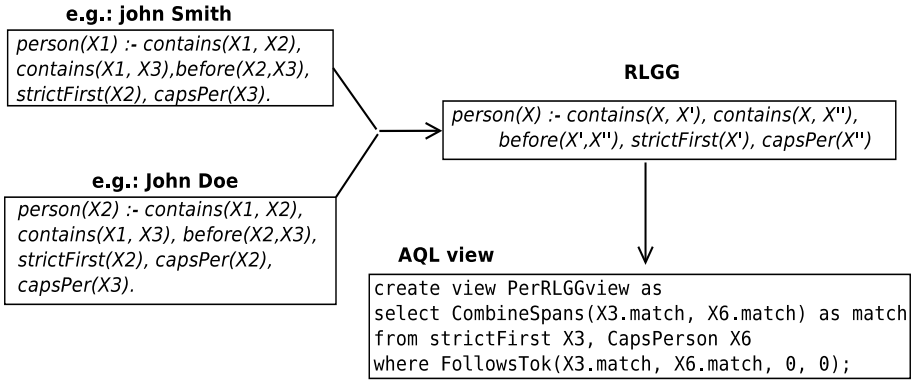


Figure 3.6: Relative Least General Generalization

1970). The bottom clause is effectively a join of the background predicates. (See Figure 3.4 for an example.) RLGG is associative, and we use this fact to compute RLGGs of sets of examples in a cluster. The RLGG of two bottom clauses as computed in our system, and its translation to an AQL view is illustrated in Figure 3.6. We filter out RLGGs generated due to sub-optimal clusters and convert the selected RLGGs into the equivalent AQL views. Each such AQL view is treated as a $\mathcal{CD}$ rule. We next discuss the process of filtering-out such sup-optimal clusters. We interchangeably refer to the RLGGs and the clusters they represent.

Iterative Clustering and RLGG filtering

Since clustering is sub-optimal, we can expect some clusters in a single run of clustering to have poor RLGGs — either in terms of complexity or precision. This is often because some clusters get fragmented. In rule learning literature there is a class of algorithms called *separate-and-conquer* (Fürnkranz, 1999), which is an iterative approach to alleviate this problem. Therefore, we use an iterative clustering approach to generate good clusters. In each iteration, we pick the clusters with the best RLGGs and remove all examples covered by those RLGGs. The best RLGGs must have precision and number of examples covered above a pre-specified threshold. The entire process of iterative clustering and selecting the best RLGGs is depicted in Figure 3.7.

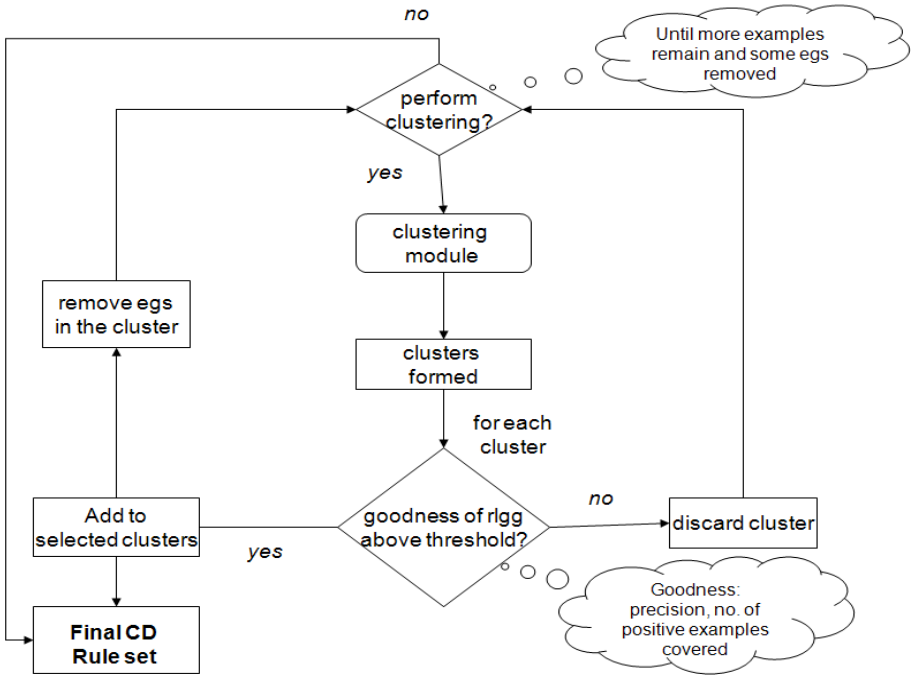


Figure 3.7: Iterative Clustering

3.4.3 Induction of Candidate-Refinement Features

Span-View Table

The $\mathcal{CD}$ features induced in phase 1 along with the textual spans they generate, yield the *span-view table*. The rows of the table correspond to the *set of spans* returned by all the CD-feature views. The columns correspond to the set of CD-feature names. Each span either belongs to one of the named entity types (PER, ORG or LOC), or is none of them (NONE); the type information constitutes its class label (see Figure 3.8 for an illustrated example). The cells in the table correspond to a match (M), or a no-match (N), or partial/overlapping match (O) of a span generated by a CD-feature view.

Propositional Rule Learning

We learn $\mathcal{CR}$ features as conjunctive compositions of the $\mathcal{CD}$ features (and their negations.) The attribute-value table discussed above is used as input to a propositional rule learner such as *JRIP* to learn such compositions of the $\mathcal{CD}$ features. We considered a number of different propositional learners and studied their features. These included both decision trees and decision list learners. Based on this study, we decided to use *RIP-*

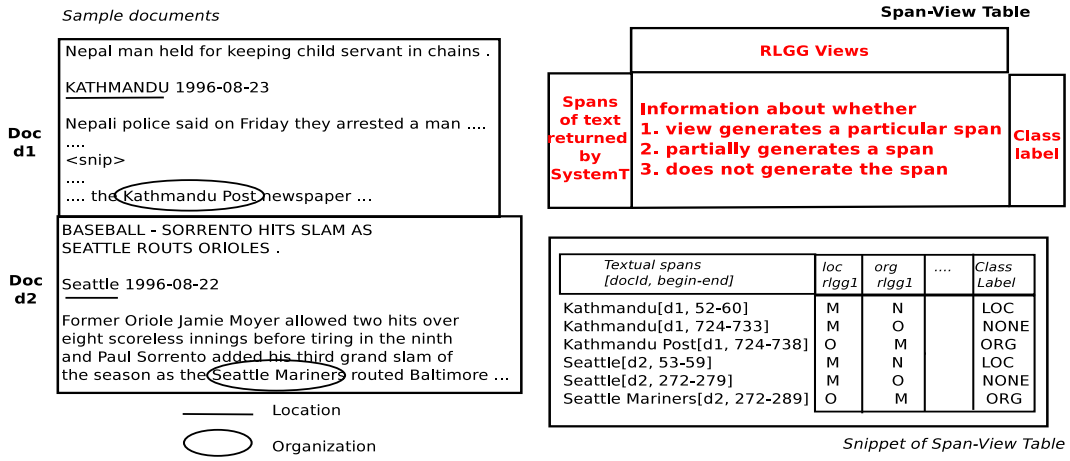


Figure 3.8: Span-View Table

PER (Fürnkranz and Widmer, 1994) implemented as the *JRIP* classifier in weka (Witten et al., 2011). Some considerations that favor JRIP are as follows:

1. Absence of rule ordering.
2. Ease of conversion to AQL.
3. Amenability to add induction biases in the implementation

The *RIPPER* algorithm starts constructing rules for one class at a time, starting with the class containing the least number of instances. The induction of a rule-set for each class consists of two phases : the *build phase* and the *optimize phase*. The build phase uses a modified version of the *incremental reduced error pruning (IREP)* algorithm (Fürnkranz and Widmer, 1994) to construct rules. It uses *minimum description length* criteria as an extra stopping condition, apart from the regular stopping criteria used in IREP. We modified JRIP to consider negation of propositions in the build phase. The optimize phase performs post-processing of the rules obtained by IREP using the conventional *reduced error pruning (REP)*. This phase tries to emulate the global-optimization strategy of REP. The RIPPER algorithm is explained in detail in Witten et al. (2011) and Cohen (1995).

Induction Bias

A number of syntactic biases were introduced in JRIP to aid in the interpretability of the induced rules. We observed in our manually developed rules that $\mathcal{CR}$ rules for a type involve interaction between $\mathcal{CD}$s for the same type and negations (not-overlaps, not matches) of $\mathcal{CD}$s of the other types. This bias was incorporated by constraining a JRIP

rule to contain only positive features ($\mathcal{CD}$s) of the same type (for instance PER), and negative features ($\mathcal{CD}$s) of only other types (ORG and LOC, in this case).

The output of the JRIP algorithm is a set of rules, one set for each of PER, ORG and LOC. Here is an example rule: `PER-CR-Rule` $\Leftarrow$ `(Per`$\mathcal{CD}$ `= m) AND (Loc`$\mathcal{CD}$ `!= o)`, which is read as : *"If a span matches Per*$\mathcal{CD}$ *and does not overlap with LocCD, then that span denotes a PER named entity."*Here `PerCD` is {`[FirstName` $\wedge$ `CapsPerson][LastName` $\wedge$ `CapsPerson]`} [7] and `LocCD` is {`[CapsPlace` $\wedge$ `CitiesDict]`}. This rule filters out wrong person annotations such as *"Prince William"* in *Prince William Sound*. (This is the name of a location but has overlapped with a person named entity.) In AQL, this effect can be achieved most elegantly by the `minus` (filter) construct. Such an AQL rule will filter all those occurrences of *Prince William* from the list of persons that overlap with a city name.

Steps such as clustering, computation of RLGGs, JRIP, and theorem-proving using SystemT were parallelized. Once the $\mathcal{CR}$ views for each type of named entity are learned, many forms of consolidations (COs) are possible, both within and across types. A simple consolidation policy that we have incorporated in the system is as follows: union all the rules of a particular type, then perform a *contained within* consolidation, which results in the final set of consolidated views for each named entity type.

3.5 Experiments

3.5.1 Experimental Setup

We evaluate our system on CoNLL03 (Tjong Kim Sang and De Meulder, 2003), a collection of Reuters news stories. We used the CoNLL03 training set for induction, and report results on the CoNLL03 test collection. In Table 3.2, we present a summary of the CoNLL03 dataset.

The basic features (BFs) form the primary input to our induction system. We experimented with three sets of BFs:

1. *Initial set(**E1**)*: The goal in this setup was to induce an initial set of rules based on a small set of reasonable BFs. We used a conservative initial set consisting of fifteen BFs (five regular expressions and ten dictionaries).

[7] Two consecutive spans where the 1st is FirstName and CapsPerson and the 2nd is LastName and CapsPerson.

Type of NE	*Number of Examples*		
	Train	*Dev*	*Test*
PER	6560	1830	1593
ORG	6312	1333	1648
LOC	7119	1828	1657

Table 3.2: Dataset details

2. *Enhanced set (**E2**)*: Based on the results of E1, we identified a set of additional domain-independent BFs[8]. Five views were added to the existing set in E1 (one regular expression and four dictionaries). The goal was to observe whether our approach yielded reasonable accuracies compared to generic rules developed manually.

3. *Domain customized set (**E3**)*: Based on the knowledge of the domain of the training dataset (CoNLL03), we introduced a set of features that was specific to this dataset. This included sports-related person, organization and location dictionaries[9]. These views were added to the existing set in E2. The intended goal was to observe what the best possible accuracies that could be achieved with BFs customized to a particular domain were.

The set of parameters for iterative clustering on which the accuracies reported were: the precision threshold for the RLGGs of the clusters was 70%, and the number of examples covered by each RLGG was five. We selected the top five clusters from each iteration, the RLGGs of which, crossed this threshold. If there were no such clusters then we would lower the precision threshold to 35% (half of the threshold). When no new clusters were formed, we ended the iterations.

	Train			Test			
Type	**P**	**R**	**F**	**P**	**R**	**F**	**C(E)**
E1 (Initial set)							
PER	88.5	41.4	56.4	92.5	39.4	55.3	144
ORG	89.1	7.3	13.4	85.9	5.2	9.7	22
LOC	91.6	54.5	68.3	87.3	55.3	67.8	105
Overall	90.2	35.3	50.7	89.2	33.3	48.5	234
E2 (Enhanced set)							
PER	84.7	52.9	65.1	87.5	49.9	63.5	233
ORG	88.2	7.8	14.3	85.8	5.9	11.0	99
LOC	92.1	58.6	71.7	88.6	59.1	70.9	257
Overall	88.6	40.7	55.8	88.0	38.2	53.3	457
E3 (Domain customized set)							
PER	89.9	57.3	70.0	91.7	56.0	69.5	430
ORG	86.9	50.9	64.2	86.9	47.5	61.4	348
LOC	90.8	67.0	77.1	84.3	67.3	74.8	356
Overall	89.4	58.7	70.9	87.3	57.0	68.9	844

Table 3.3: Results on CoNLL03 dataset with different basic feature sets.

3.5.2 Experiments and Results

Effect of Augmenting Basic Features

Table 3.3 shows the accuracy and complexity of rules induced with the three basic feature sets E1, E2 and E3, respectively [10]. The overall F-measure on the test dataset is 48.5% with E1; it increases to around 53.3% with E2, and is highest at 68.9% with E3. As we increase the number of BFs, the accuracies of the induced extractors increase, at the cost of an increase in complexity. In particular, the recall increases significantly across the board, and is more prominent between E2 and E3, where the additional domain-specific

[8]For example, the feature *preposition dictionary* was added in E2 to help identify organization names such as *Bank of England*.

[9]Half of the documents in CoNLL03 are sports-related.

[10]These are the results for the configuration with bias.

features result in recall increase from 5.9% to 47.5% for ORG. The precision increases slightly for PER, but decreases slightly for LOC and ORG with the addition of domain specific-features.

Comparison with manually developed rules

We compared the induced extractors with the manually developed extractors of Chiticariu et al. (2010b), heretofore referred to as *manual extractors*. (For a detailed analysis, we obtained the extractors from the authors). Table 3.4 shows the accuracy and complexity of the induced rules with E2 and E3 and the manual extractors for the generic domain and, respectively, customized for the CoNLL03 domain. (In the table, ignore the column *Induced (without bias)*, which is discussed later). Our technique compares reasonably with the manually constructed generic extractor for two of the three entity types, and on precision, for all entity types. This is especially so because our system generated the rules in one hour, whereas, the development of manual rules took much longer [11]. Additional work is required to match the manual customized extractor's performance, primarily due to shortcomings in our current target language. Recall that our framework is limited to a small subset of AQL constructs for expressing $\mathcal{CD}$ and $\mathcal{CR}$ rules, and there is a single consolidation rule. In particular, advanced constructs such as dynamic dictionaries are not supported, and the set of predicates to the Filter construct supported in our system is restricted to predicates over other concepts, which is only a subset of those used in Chiticariu et al. (2010b). The manual extractors also contain a larger number of rules that cover many different cases, which improves the accuracy, but also leads to a higher complexity score. To better analyze the complexity, we also computed the average rule length for each extractor by dividing the complexity score by the number of AQL views of the extractor. The average rule length is 1.78 and 1.87 for the induced extractors with E2 and E3, respectively, and 1.9 and 2.1 for the generic and customized extractors of Chiticariu et al. (2010b), respectively. The average rule length increases from the generic extractor to the customized extractor in both cases. On average, however, an individual induced rule is slightly smaller than a manually developed rule.

[11] Chiticariu et al. (2010b) mention that customization for three domains required eight person weeks. It is reasonable to infer that developing the generic rules took comparable effort.

Effect of Bias

The goal of this experiment is to demonstrate the importance of biases in the induction process. The biases added to the system are broadly of two types: (i) Partition of basic features based on types (ii) Restriction on the type of $\mathcal{CD}$ views that can appear in a $\mathcal{CR}$ view. [12] Without the partition of features in (i) many semantically similar basic features (especially, regular expressions) would match a given token, leading to an increase in the length of a $\mathcal{CD}$ rule. For example, in the $\mathcal{CD}$ rule [FirstNameDict][CapsPerson $\wedge$ CapsOrg]} (*"A FirstNameDict span followed by a CapsPerson span that is also a CapsOrg span"*), CapsPerson and CapsOrg are two very similar regular expressions that identify capitalized phrases that look like person, and respectively, organization names, with small variations (for example, the former may allow special characters such as '-'.) Including both BFs in a $\mathcal{CD}$ rule leads to a larger rule that is unintuitive for a developer. The former bias excludes such $\mathcal{CD}$ rules from consideration.

The latter type of bias prevents $\mathcal{CD}$ rules of one type to appear as positive clues for a $\mathcal{CR}$ rule of a different type. For instance, without this bias, one of the $\mathcal{CR}$ rules obtained was `Per` $\Leftarrow$ `(Org`$\mathcal{CD}$ `= m) AND (Loc`$\mathcal{CD}$ `!= o)` (*"If a span matches Org*$\mathcal{CD}$ *and does not overlap with LocCD, then that span denotes a PER named entity"*.) Here, `OrgCD` was {`[CapsOrg][CapsOrg]`} and `LocCD` was {`[CapsLoc` $\wedge$ `CitiesDict]`}. The inclusion of an Organization $\mathcal{CD}$ rule as a positive clue for a Person $\mathcal{CR}$ rule is unintuitive for a developer.

Table 3.4, shows the effect (for E2 and E3) of disabling and enabling bias on the test dataset during the induction of $\mathcal{CR}$ rules using JRIP. Adding bias improves the precision of the induced rules. Without bias, however, the system is less constrained in its search for high recall rules, leading to a slightly higher overall F measure. This comes at the cost of an increase in extractor complexity and average rule length. For example, for E2, the average rule length decreases from 2.17 to 1.78 after adding the bias. Overall, our results show that biases lead to less complex extractors, with only a very minor effect on accuracy. Thus, biases are important factors that contribute to inducing rules that are understandable and may be refined by humans.

[12] For example, person $\mathcal{CR}$ view can contain only person $\mathcal{CD}$ views as positive clues, and $\mathcal{CD}$ views of other types as negative clues.

		Chiticariu et al. 2010b				Induced (With Bias)				Induced (Without Bias)			
		P	**R**	**F**	**C(E)**	**P**	**R**	**F**	**C(E)**	**P**	**R**	**F**	**C(E)**
Generic (E2)	*PER*	82.2	60.3	69.5	945	87.5	49.9	63.5	233	85.8	53.7	66.0	476
	ORG	75.7	17.5	28.5	1015	85.8	5.9	11.0	99	74.1	15.7	25.9	327
	LOC	72.2	86.1	78.6	921	88.6	59.1	70.9	257	85.9	61.5	71.7	303
	Overall	75.9	54.6	63.5	1015	88.0	38.2	53.3	457	84.2	43.5	57.4	907
Customized (E3)	*PER*	96.3	92.2	94.2	2154	91.7	56.0	69.5	430	90.7	60.3	72.4	359
	ORG	91.1	85.1	88.0	2154	86.9	47.5	61.4	348	90.4	46.8	61.7	397
	LOC	93.3	91.7	92.5	2154	84.3	67.3	74.8	356	83.9	69.1	75.8	486
	Overall	93.5	89.6	91.5	2160	87.3	57.0	68.9	844	87.8	58.7	70.4	901

Table 3.4: Comparison of induced rules (with and without bias) and manually developed rules. (CoNLL03 test dataset)

Comparison with other induction systems

We also experimented with two other induction systems, Aleph[13] and ALP[14], a package that implements one of the reportedly good information extraction algorithms (Ciravegna, 2001). While induction in Aleph was performed with the same target language as in our approach, the target language of ALP is JAPE, which has been shown by Chiticariu et al. (2010b) to lack in some of the constructs (such as minus) that AQL provides and which form a part of our target language (especially the rule refinement phase). However, despite experimenting with all possible parameter configurations for each of these (in each of E1, E2 and E3 settings), the accuracies obtained were substantially (30-50%) inferior and the extractor complexity was much (around 60%) higher, when compared to our system (with or without bias). Additionally, Aleph takes close to three days for induction, whereas, both ALP and our system require less than an hour.

[13]A system for inductive logic programming. See http://www.cs.ox.ac.uk/activities/machlearn/Aleph/aleph.html

[14]http://code.google.com/p/alpie/

3.5.3 Discussion

Weak and Strong $\mathcal{CD}$s reflected in $\mathcal{CR}$s

In our experiments, we found that varying the precision and complexity thresholds while inducing the $\mathcal{CD}$s(*c.f* Section 3.4) affected the F1 of the final extractor only minimally. However, reducing the precision threshold generally improved the precision of the final extractor, which seemed counter-intuitive at first. We found that $\mathcal{CR}$ rules learned by JRIP consist of a strong $\mathcal{CD}$ rule (high precision, typically involving a dictionary) and a weak $\mathcal{CD}$ rule (low precision, typically involving only regular expressions). The strong $\mathcal{CD}$ rule always corresponded to a positive clue (match) and the weak $\mathcal{CD}$ rule corresponded to the negative clue (overlaps or not-matches). This is illustrated in the following $\mathcal{CR}$ rule: `PER` $\Leftarrow$ `(Per`$\mathcal{CD}$ `= m) AND (Org`$\mathcal{CD}$ `!= o)` where `(PerCD` is {`[CapsPersonR] [CapsPersonR` $\wedge$ `LastNameDict]`} and `(OrgCD` is {`[CapsOrgR][CapsOrgR][CapsOrgR]`}. This is posited to be the way the $\mathcal{CR}$ rule learner operates — it tries to learn conjunctions of weak and strong clues so as to filter one from the other. Therefore, setting a precision threshold too high limited the number of such weak clues and the ability of the $\mathcal{CR}$ rule learner to find such rules.

Interpretability

Measuring interpretability of rules is a difficult problem. In this work, we have taken the first step toward measuring interpretability using a coarse grain measure in the form of a simple notion of complexity score. The complexity is very helpful in comparing alternative rule sets based on the number of rules and the size of each rule, but exhibits a number of shortcomings, described next. First, it disregards other components of a rule besides its from clause, for example, the number of items in the select clause, or the where clause. Second, rule developers use semantically meaningful view names, such as those shown in Figure B.1 to help them recall the semantics of a rule at a high-level — an aspect that is not captured by the complexity measure. Automatic generation of meaningful names for induced views is an interesting direction for future work. Finally, the overall structure of an extractor is not considered. In simple terms, an extractor that consists of five rules of size one is indistinguishable from an extractor that consists of a single rule of size five; which of these extractors is more interpretable is arguable. More generally, the extent of this shortcoming is best explained using an example. While informally examining the

rules induced by our system, we found that $\mathcal{CD}$ rules are similar in spirit to those written by rule developers. On the other hand, the induced $\mathcal{CR}$ rules are too fine-grained. In general, rule developers group $\mathcal{CD}$ rules with similar semantics, then write refinement rules at the higher level of the group, as opposed to the lower level of individual $\mathcal{CD}$ views. For example, one may write multiple $\mathcal{CD}$ rules for candidate person names of the form $\langle First \rangle \langle Last \rangle$, and multiple $\mathcal{CD}$ rules of the form $\langle Last \rangle$, $\langle First \rangle$. One would then union together the candidates from each of the two groups into two different views, for example, *PerFirstLast* and *PerLastCommaFirst*, and write filter rules at the higher level of these two views, for example, *"Remove PerLastCommaFirst spans that overlap with a PerFirstLast span"*. In contrast, our induction algorithm considers $\mathcal{CR}$ rules consisting of combinations of $\mathcal{CD}$ rules directly, leading to many semantically similar $\mathcal{CR}$ rules, each operating over small parts of a larger semantic group (see rule in Section 3.5.2). This results in repetition, and qualitatively less interpretable rules, since humans prefer higher levels of abstraction and generalization. This nuance is not captured by the complexity score which may deem an extractor consisting of many rules, where many of the rules operate at higher levels of groups of candidates, to be more complex than a smaller extractor with many fine-grained rules. Indeed, as shown before, the complexity of the induced extractors is much smaller compared to that of manual extractors, although, the latter follow the semantic grouping principle and are considered more interpretable.

3.6 Relational Features in a Max-margin Setting

In this section, we describe our attempts to use relational features in max-margin based learning algorithms. We model the named-entity recognition problem as a sequence-labeling task and jointly learn the class labels in this structured output space.

3.6.1 Sequence Labeling

The objective in sequence labeling is to assign a state (class label) to every instance of a sequence of observations (inputs). For example, figure 3.9 depicts the named-entity recognition problem as a sequence labeling task.

Typical sequence-labeling algorithms learn probabilistic information about (transition) relationships between neighboring states along with probabilistic information about the (emission/observation) relationships between the states and observations.

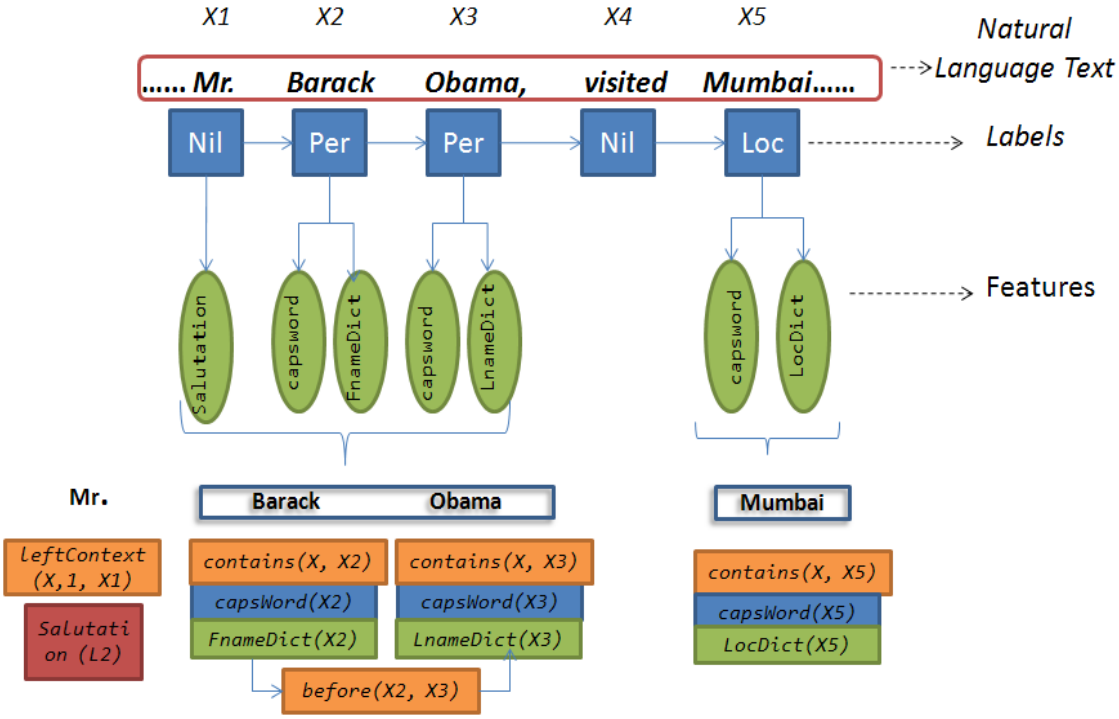


Figure 3.9: NER as Sequence Labeling.

Hidden Markov Models (HMM) (Rabiner, 1990), Conditional Random Fields (CRF) (Lafferty et al., 2001), and Support Vector Machines on Structured Output Spaces StructSVM (Tsochantaridis et al., 2004) are three popularly used models for sequence-labeling problems. The probabilistic information is later used to identify the (hidden) label sequence that best explains the given sequence of observations.

3.6.2 Models for Sequence Labeling

In sequence-labeling models, the objective is to learn functions of the form $\mathcal{F} : \mathcal{X} \rightarrow \mathcal{Y}$ from the training data, where $\mathcal{X}$ and $\mathcal{Y}$ are input and output sequence spaces, respectively. Typically, a discriminant function $F : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ is learned from training data that consists of pairs of input and output sequences. The prediction is performed using the decision function $\mathcal{F}(X; \mathbf{f}) = \arg\max_{Y \in \mathcal{Y}} F(X, Y; \mathbf{f})$. $F(X, Y; \mathbf{f}) = \langle \mathbf{f}, \boldsymbol{\psi}(X, Y) \rangle$ represents a score which is a scalar value based on the features $\boldsymbol{\psi}$ involving the values of input sequence X and output sequence Y and parameterized by a parameter vector $\mathbf{f}$. In sequence prediction, features are constructed to represent emission (observation) and transition distributions. Given this objective, we can classify sequence labeling techniques into

probability-based and max-margin based, which we discuss in the following paragraphs.

Probability-based methods

Here the parameters are characterized by probabilities. Hidden Markov Models (HMM) (Rabiner, 1990) and Conditional Random Fields (CRF) (Lafferty et al., 2001) are traditionally used in sequence-prediction problems and have an objective that is similar to the one discussed above — with probabilities and potential weights as parameters, respectively. Their ability to capture the state transition dependencies along with the observation dependencies makes these approaches robust in noisy and sparse data.

In an HMM setup, probability parameters that maximize the joint probability $p(X, Y)$ of input training sequence X and output training sequence Y are learned during the training phase. From the independence assumptions in an HMM, one can factorize the joint probability distribution of the sequence of inputs (X) and labels (Y) into three factors: the initial state distribution $p(y^1)$, the transition distribution $p(y^t|y^{t-1})$, and the emission distribution $p(x^t|y^t)$ (Rabiner, 1990). Here x^t and y^t represent the input variable and the class variable at position t in the sequence, respectively. Therefore, $p(X, Y) = \prod_{t=1}^{T} p(y^t|y^{t-1})p(x^t|y^t)$, where T is the length of the sequence and $p(y^1|y^0)$ is used instead of $p(y^1)$ to simplify notation. Parameters for the distributions are learned by maximizing $p(X, Y)$. In contrast, CRF (Lafferty et al., 2001) learns parameters that maximize $p(Y|X)$ — the conditional probability of a sequence of states Y given a sequence of inputs X — where, $p(Y|X) = \frac{1}{Z(X)}\ exp \sum_{t=1}^{T} \phi_t(y^t, X) + \phi_{t-1}(y^{t-1}, y^t, X)$, in which, $\phi_t(y^t, X)$ and $\phi_{t-1}(y^{t-1}, y^t, X)$ stands for potential functions and $Z(X)$ is the partition function. These parameters are later used to identify the (hidden) label sequence that best explains a given sequence of inputs (or observations) — that is usually performed by a dynamic programming algorithm known as the Viterbi Algorithm (Forney, 1973).

Max-margin based methods

StructSVM. Tsochantaridis et al. (2004) generalize the SVM framework to perform classification on structured outputs. This builds on the conventional SVM formulation that assumes the output to be a single variable which can be a binary label or multi-class. StructSVM generalizes multi-class Support Vector Machine learning to incorporate

features constructed from input and output variables, and solves classification problems with structured output data. The loss function in sequence labeling has to be chosen such that the predicted sequence of labels that differ from the actual labels in a few time steps should be penalized lesser than those that differ from the actual labels in a majority of sequence positions. A loss function is represented as $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$. $\Delta(Y, \hat{Y})$ is the loss value when the true output is Y and the prediction is $\hat{Y}$. The SVM formulation for structured output spaces can be written as,

$$\min_{\mathbf{f},\boldsymbol{\xi}} \frac{1}{2} \parallel \mathbf{f} \parallel^2 + \frac{C}{m} \sum_{i=1}^{m} \xi_i, \qquad s.t. \ \forall i : \ \xi_i \geq 0$$
$$\forall i, \ \forall \, Y \in \mathcal{Y} \setminus Y_i : \ \langle \mathbf{f}, \boldsymbol{\psi}_i^{\delta}(Y) \rangle \geq 1 - \frac{\xi_i}{\Delta(Y_i, Y)}. \tag{3.1}$$

where m is the number of examples, C is the regularization parameter, ξ's are the slack variables introduced to allow errors in the training set in a soft margin SVM formulation, $X_i \in \mathcal{X}$ and $Y_i \in \mathcal{Y}$ represent the i^{th} input and output sequences, respectively and $\langle \mathbf{f}, \boldsymbol{\psi}_i^{\delta}(Y) \rangle$ represents the value $\langle \mathbf{f}, \boldsymbol{\psi}(X_i, Y_i) \rangle - \langle \mathbf{f}, \boldsymbol{\psi}(X_i, Y) \rangle$. In cases where the sequence length is large, the number of constraints in (3.1) can be extremely large. To solve this problem, an algorithm based on the cutting plane method is proposed (*c.f.* algorithm 1 in Tsochantaridis et al. (2004)) in order to find a polynomially sized subset of constraints that ensures a solution very close to the optimum.

RELHKL and StructHKL. Jawanpuria et al. (2011) propose Rule Ensemble Learning using Hierarchical Kernels (RELHKL), in which they make use of the Hierarchical Kernel Learning (HKL) framework introduced in Bach (2009) to simultaneously learn sparse rule ensembles and their optimal weights for binary classification problems. They use a hierarchical ρ-norm regularizer to select a sparse set of features from an exponential space of features. The prime objective of Rule Ensemble Learning (REL) is to learn small set of simple rules and their optimal weights. The set of rules that can be constructed from basic features follow a partial order and can be visualized as a lattice (conjunction lattice, when the features are conjunctions of basic features.) The set of indices of the nodes in the lattice are represented by $\mathcal{V}$. To learn sparse sets of rules, the regularizer $\Omega(\mathbf{f})$ is defined as, $\Omega(\mathbf{f}) = \sum_{v \in \mathcal{V}} d_v \parallel \mathbf{f}_{D(v)} \parallel_{\rho}$, where $\mathbf{f}$ is the feature weight vector that corresponds to the feature nodes in the lattice, $d_v \geq 0$ is a prior parameter showing the usefulness of the feature conjunctions, $\mathbf{f}_{D(v)}$ is the vector with elements as $\parallel f_w \parallel_2 \ \forall w \in D(v)$, $D(v)$ the set of descendant nodes of v, and $\parallel . \parallel_{\rho}$ represents the ρ-norm. In rule ensemble learning,

d_v is defined as $\beta^{|v|}$, where β is a constant. Here, the 1-norm over the sub lattices formed by the descendants of nodes helps to select only a few sub lattices and the ρ-norm ensures sparsity among the selected sub lattices. Since only a few features are expected to be non-zero at optimality, for computational efficiency, an active set algorithm is employed (*c.f.* algorithm 1 of Jawanpuria et al. (2011)).

The RELHKL approach is specific to the single variable binary classification problem and cannot be trivially applied to problems that involve multi-class structured output data. Nair et al. (2012b), present a generalization of the RELHKL formulation for structured output spaces (StructHKL) and discuss how interesting relational features can be learned using that new framework. StructHKL optimally and efficiently learns discriminative features for multi-class structured output classification problems such as sequence labeling. They build on the StructSVM framework for sequence-prediction problems, in which all possible $\mathcal{SC}$s form the input features while the transition features are constructed from all possible transitions between state labels. They use a ρ-norm hierarchical regularizer to select a sparse set of $\mathcal{SC}$s as emission features.The exponentially large observation feature space is searched using an active-set algorithm, and the exponentially large set of constraints is handled using a cutting-plane algorithm.

3.6.3 Experiments

We conducted our experiments on the same CoNLL03 dataset as discussed in section 3.5.2. There are three major aspects in the design of an NER system.

1. Features: These include gazetteers, regular expressions, lexical features and so on. State-of-the-art NER systems use a very large collection of features. Our goal in this section is to perform a comparative study of the use of relational features in an out-of-the-box ML system. So we use the same set of a modest number of features described in our earlier experimental setup (E1).

2. Language of the features: This is related to the feature space categorization that we discussed in chapter 2. These can be termed as feature templates. Our features belong to BFs, $\mathcal{SC}$s and $\mathcal{CD}$s in our experiments here.

3. Learning algorithm: There are a number of choices for learning algorithms including

	Overall			PER			ORG			LOC		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
`BFs on StructSVM`	24.44	5.32	8.74	25.94	4.68	7.92	18.74	4.16	6.81	26.84	6.82	10.87
`BFs on StructHKL`	13.25	6.42	8.65	10.31	9.54	9.91	9.09	0.33	0.64	21.96	7.46	11.14
$\mathcal{CD}$`s on StructSVM`	31.41	8.84	13.79	37.64	14.70	21.15	23.34	9.30	13.30	28.17	2.38	4.39
`Induced Rules`	89.20	33.30	48.50	92.50	39.40	55.30	85.90	5.20	9.70	87.30	55.30	67.80

Table 3.5: Induced Features in Max-margin algorithm

CRF, HMM, StructSVM and so on. We used the StructSVM implementation [15] of Tsochantaridis et al. (2005) and the StructHKL implementation of Nair et al. (2012b) for our experiments.

The various configurations of the experiments are explained below:

1. `BFs on StructSVM`: This forms the baseline of our investigations.
2. `BFs on StructHKL`: The goal was to leverage StructHKL to learn combinations of features in the space of SCs.
3. $\mathcal{CD}$`s on StructSVM`: Here the relational $\mathcal{CD}$ features were presented to the StructSVM. The goal was to observe whether the use of $\mathcal{CD}$ features improves the baseline accuracies.
4. `Induced Rules`: The results of the rules induced using our rule-induction techniques for the same set of basic features.

The results are presented in Table 3.5. We observe an increase of $\sim$58% in the F_1 score when $\mathcal{CD}$s are used as features compared to BFs. So the use of more expressive features such as $\mathcal{CD}$s definitely provides better signal to the learning algorithm than the use of simple BFs.

The StructHKL experiments were carried out in order to automatically generate basic feature conjunctions. The space of induced feature is SCs. The results are comparable to the baseline (marginally worse overall F_1 score). We posit that $\mathcal{SC}$s are conjunctions of BFs at a single sequence position. So, learning conjuncts in this space will not result in useful generalizations because many named-entities are multi-word expressions.

[15] Available at `http://www.cs.cornell.edu/People/tj/svm_light/svm_hmm.html`

We observe that the results of the max-margin based algorithms are poor compared to the induced rules. The best F_1 score of the max-margin setting, $\mathcal{CD}$`s on StructSVM`, is 34.71% lesser than the induced rules. In the following paragraphs, we discuss some of the possible reasons for such a poor performance.

Discussion

As seen in Table 3.5, the max-margin experiments exhibit a significant drop in performance compared to rule induction. A few of the reasons for this poor performance are the subject of discussion in this section.

As discussed in Section 3.4, we use JRIP to combine $\mathcal{CD}$ features to learn more expressive feature combinations. JRIP is amenable to addition of biases, which in turn aides good feature selection. In the max-margin based sequence labeling methods, we could not find an easy way to incorporate such biases (such as the partition of $\mathcal{CD}$ features that belong to different types.) This probably hurts the performance.

There is a significant skew in the label space of the CoNLL03 dataset. The named-entity labels are very few when compared to the *nil* label. Generally, in a joint labeling task, the class imbalance problem leads to poor performance. State-of-the-art sequence labeling models for NER circumvent this problem by using a large number of features. Since our focus is on the modeling exercise rather than feature engineering, we used a small set of features.

In another sequence labeling task known as *activity recognition*, the performance of some of these max-margin models are better than the baseline. The datasets used as benchmarks in the activity recognition task have a lesser skew in the class label distribution. This might be one of the reasons for the better performance of such models. Some of these experiments and results are discussed in our previous work (Nair et al., 2012a; Nagesh et al., 2013) and other related papers (Nair et al., 2012b, 2011).

3.7 Chapter Summary

In this chapter, we presented a system for efficiently inducing named-entity annotation rules in AQL. The design of our approach is aimed at producing accurate rules that can be understood and refined by humans. A special emphasis is laid on low complexity and efficient computation of the induced rules, while mimicking a four stage approach used for

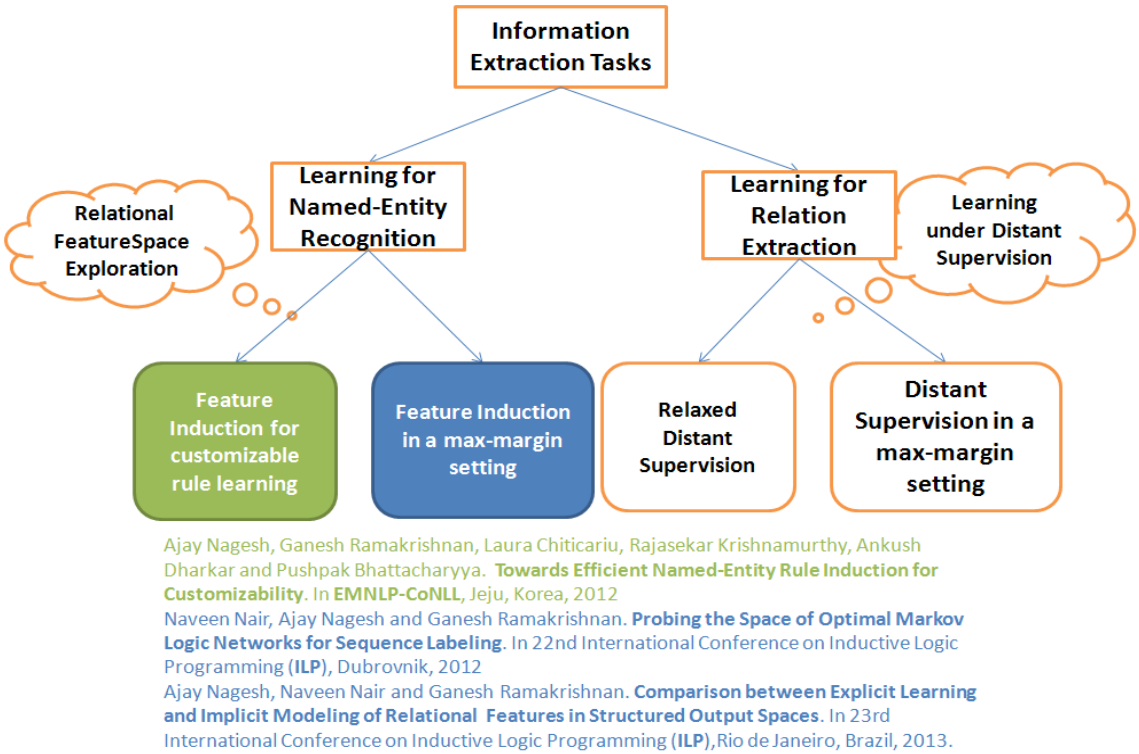


Figure 3.10: Thesis Organization: Chapter 3

manually constructing rules. The main induction approach consist of two stages, namely, candidate definition and candidate refinement, while the other two stages, namely, basic feature definition and rule consolidation are assumed to be manually specified. We presented results with both domain-independent as well customized basic features. The induced rules have good accuracy and low complexity according to our complexity measure.

In a follow-up work, we investigated the utility of relational features in max-margin based sequence-labeling models. The results in these settings have not been very promising due to a number of reasons, some of them highlighted in Section 3.6.3. Figure 3.10 highlights the part presented in this chapter (except Section 3.6) and the accompanying publication that overlaps with the material presented.

Part II

Learning for Relation Extraction

Chapter 4

Relaxed Distant Supervision

In this chapter, we will shift the gear slightly to discuss another important task in IE, namely, *relation extraction*. The task of extracting relational facts that pertain to a set of entities from natural language text is termed as *relation extraction*. For example, from a sentence in natural language sentence, such as, *"President Barack Obama on Friday defended his administration's mass collection of telephone and Internet records in the United States,"* we can infer the relation, `presidentOf(Barack Obama, United States)` between the entities ``Barack Obama'' and ``United States''.

4.1 Relation Extraction through Distant Supervision

Supervised approaches to relation extraction (GuoDong et al., 2005; Surdeanu and Ciaramita, 2007) achieve very high accuracies. However, these approaches do not scale as they are data-intensive and the cost of creating annotated data is quite high. To alleviate this problem, Mintz et al. (2009) proposed relation extraction in the paradigm of *distant supervision*. In this approach – given a database of facts (for example, Freebase[1]) and an unannotated document collection – the goal is to heuristically align the facts in the database to the sentences in the corpus that contains the entities mentioned in a fact. This is done to create weakly labeled training data to train a classifier for relation extraction. Figure 4.1 illustrates the distant supervision paradigm with an example.

Some comparison of training data sizes: The benchmark dataset for relation extraction ACE contains $17,000$ relation instances that belong to 23 relations. In contrast, if we consider a snapshot of Freebase, it contains around 1.8 million relation instances that belong to 102 relations that connect around $940,000$ entities. Consequently, the

[1] `www.freebase.com`

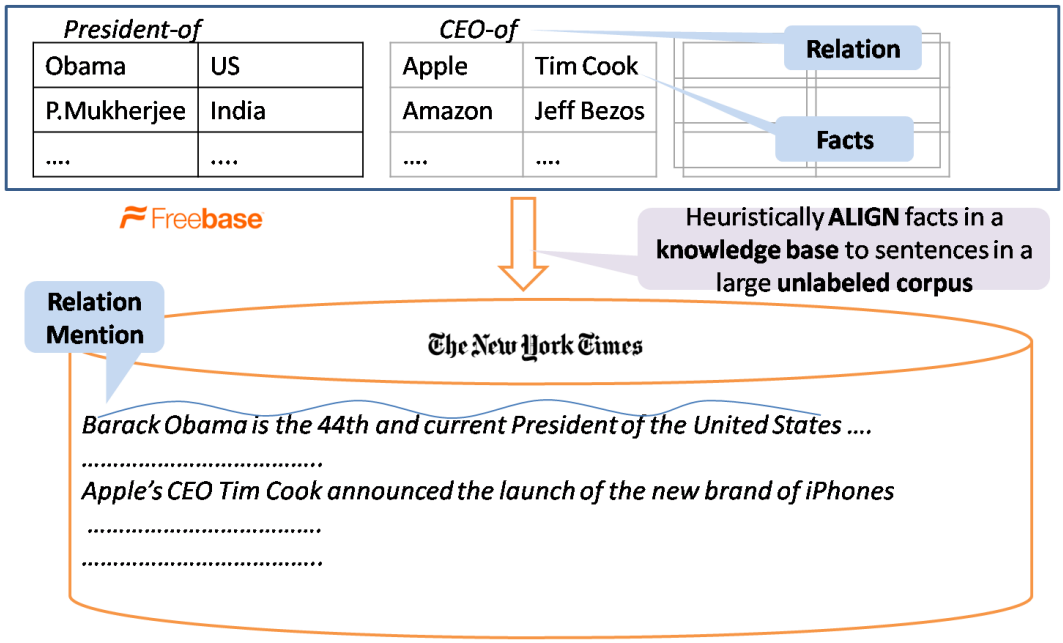


Figure 4.1: Distant Supervision for Relation Extraction

size of the data for the training of relation extractors increases manifold. However, the training dataset created by this alignment process is very noisy in nature. This has been elaborated subsequently in this chapter.

In the model proposed by Mintz et al. (2009), the underlying assumption is that *all mentions of an entity pair*[2] *(that is, sentences the contain the entity pair) in the corpus express the same relation as that which is stated in the database.* This assumption is a weak one and is often violated in natural language text. For instance, the entity pair (`Barack Obama`, `United States`) participates in more than one relation: `citizenOf`, `presidentOf`, `bornIn`; every mention expresses either one of these fixed set of relations or none of them. The important aspect to note is that every mention of the entity pair mentioned above does not express the `presidentOf` relationship. For example, *"Barack Obama was born in Honolulu, Hawaii, United States"* expresses the `bornIn` relation and *"President Barack Obama left United States this Saturday for a UN summit in Geneva"* does not express any of the relations mentioned above.

[2]In this work we restrict ourselves to binary relations

4.1.1 Related Work

Consequently, a number of models have been proposed in literature to provide better heuristics for the mapping between the entity pair in the database and its mentions in the sentences of the corpus. Riedel et al. (2010) tighten the assumption of distant supervision in the following manner: "Given a pair of entities and their mentions in sentences from a corpus, *at least one* of the mentions expresses the relation given in the database." In other words, it models the problem as multi-instance (mentions) single-label (relation) learning. The work by Yao et al. (2010) is in the same setting. However, it additionally models the selectional preferences of the arguments of participating entities in a relation. For instance, in the `presidentOf` relation, the first argument should be of type `person` and the second argument should be of type `country`. Following this, Hoffmann et al. (2011) and Surdeanu et al. (2012) propose models that consider the mapping as multi-instance multi-label learning. The instances are the mentions of the entity pair in the sentences of the corpus; also, the entity pair can participate in more than one relation. In the following subsections, we present a more detailed summary of these works.

Distant supervision for relation extraction without labeled data

As mentioned earlier, one of the first papers that introduced the paradigm of distant supervision for relation extraction was Mintz et al. (2009). They have a simplistic assumption that *all* sentences that mention an entity pair express the relation mentioned in the knowledge base. They combine the features present in all the sentences of an entity pair into one feature vector. The rationale for this type of approach is that features present in one sentence offer a noisy signal at best to determine the relation expressed by them. This combination of noisy features will make the model more robust. They train a multi-class logistic regression classifier to learn the weights of each noisy feature. Through experiments, the authors argue that syntactic features help more than lexical features. This is especially true in cases where the related entities are not lexically close but are close to each other in the dependency path of the sentence.

Modeling Relations and their Mentions without Labeled Text

Riedel et al. (2010) introduced the *at-least-once* assumption as an improvement over the simplistic model of Mintz et al. (2009). To reiterate, previous work assumed that all sentences that contain an entity pair express the relation mentioned in Freebase for that pair. However this might not always be true. For example, `nationality(Obama, USA)` might be a relation in Freebase; however, in news articles, all sentences that contain `Obama` and `USA` might not actually express the `nationality` because he is also the `President`. Therefore, the paper makes a claim that *at least one* of the sentences in the corpus expresses the relation rather than *all* the sentences. They model the *at least one constraint* as a factor graph that has latent variables for relation and relation mention. The parameters of the model are learned using a technique called *SampleRank*, which is a version of constraint-driven semi-supervised learning.

Collective Cross-Document Relation Extraction without Labeled Data

Another important work in the distant supervision literature is that of Yao et al. (2010). The work models a crucial aspect of relation discovery, namely, selection preferences of the relation and the participating entities. For instance, in the `nationality` relation, the first entity has to be of type `person` and the second entity has to be of type `country` (not any arbitrary `location` type.) It jointly models the selection preference of the relation and the participating entities (T_join and T_pair). The distant supervision assumption is that of Mintz et al. (2009). The labels for the entities (finer-grained entity types such as `founder, citizen` from course entity types such as `Person`) are obtained from Freebase. They also use a bias template (T_bias) to compute the prior of the entity types, and mention template (T_men) to link a pair of entities (relation latent variable) to its corresponding mentions in text. They employ a *joint-model* of the selection preferences and contrast it with a *pipeline model* of relations conditioned on entity types that are predicted previously in the pipeline. An NER tagger is used to obtain the coarse-grained entity mentions. They use fifty-four most frequently mentioned relations and the corresponding ten fine-grained entity types.

Knowledge-based Weak Supervision for Information Extraction of Overlapping Relations

Hoffmann et al. (2011) present a multi-instance learning algorithm (similar to Riedel et al. (2010)). However, it additionally considers overlapping relations between entities and does not treat relations as disjoint. For instance, it models the fact that between the entities `Jobs` and `Apple`, more than one relation label can be true, for instance, `CEO(Jobs, Apple)` and `Founded(Jobs, Apple)`. The key modeling differences between this work and that of Riedel et al. (2010) are: i) The sentence level variables are multi-valued compared to binary variables; ii) The aggregate relation variables are linked to model overlap compared to a single aggregate relation variable; and iii) The relation variables are joined by a *deterministic-or* compared to an aggregation of features. The authors motivate by providing some statistics of interest: 18.3% of weak-supervision facts have more than one relation. The model they propose is an undirected graphical model that allows joint reasoning about aggregate (corpus-level) and sentence level extraction decisions. The learning algorithm has perceptron-style parameter updates with two modifications: i) online learning ii) Viterbi approximation. The inference is cast as solving *a weighted edge-cover* problem. We discuss this in detail in Section 4.2 because our work is based on this model.

Multi-instance Multi-label Learning for Relation Extraction

Surdeanu et al. (2012) also present a state-of-the-art model for distant supervision-based relation extraction. Similar to Hoffmann et al. (2011), the problem is posed as a multi-instance (many instances of sentences of a pair of entities in the corpus) multi-label (a pair of entities can take multiple labels) learning problem. It has a latent variable for every sentence that contains a pair of entities (mention). It trains a multi-class classifier over these variables. It has k binary classifiers for each of the relations. It uses an EM-style algorithm for training. In the E-step the latent mention variables are discovered. In the M-step, the *likelihood* of the data given the current assignments of latent variables is maximized.

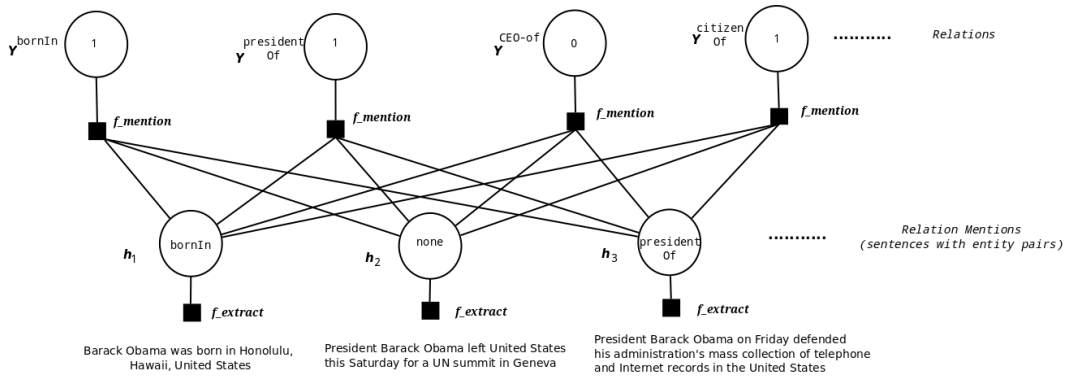


Figure 4.2: Graphical model instantiated for an entity-pair: Hoffmann et al. (2011)

4.2 Modeling Constraints in Distant Supervision

Although, the models presented in Section 4.1.1 work very well in practice, they have a number of shortcomings. One of them is the possibility that during the alignment, a fact in the database might not have an instantiation in the corpus. For instance, if our corpus only contains documents from the years 2000 to 2005, the fact `presidentOf(Barack Obama, United States)` will not be present in the corpus. In such cases, the distant supervision assumption fails to provide a mapping for the fact in the corpus.

In this chapter, we address this situation with a *noisy-or* model (Srinivas, 1993) in training the relation extractor by relaxing the *"at least one"* assumption discussed above. Our research contributions are the following:

1. We formulate the inference procedures in the training algorithm as integer linear programming (*ILP*) problems.

2. We introduce a soft constraint in the ILP objective to model noisy-or in training.

3. Empirically, our algorithm performs better than the procedure by Hoffmann et al. (2011) under certain settings on two benchmark datasets.

Our work extends the work of Hoffmann et al. (2011). Therefore, we recapitulate Hoffmann's model in the following subsection, following which, our additions to this model are explained in detail.

Hoffmann's model

As mentioned previously, Hoffmann et al. (2011) present a multi-instance multi-label model for relation extraction through distant supervision. In this model, a pair of entities

has multiple mentions (sentence containing the entity pair) in the corpus. An entity pair can have one or more relation labels (obtained from the database). An example instantiation for the entity pair `<Barack Obama, United States>` is shown in Figure 4.2. The inter-dependencies between relation labels and (hidden) mention labels are modeled by a Markov Random Field (Figure 4.2).

Objective function

Consider an entity pair (e_1, e_2) denoted by the index i. The set of sentences containing the entity pair is denoted $\mathbf{x_i}$, and the set of relation labels for the entity pair from the database is denoted by $\mathbf{y_i}$. The mention-level labels are denoted by the latent variable $\mathbf{h}$ (there is one variable h_j for each sentence j). To learn the parameters θ, the training objective to maximize is the likelihood of the facts observed in the database conditioned on the sentences in the text corpus.

$$\theta^* = \arg\max_{\theta} \prod_i Pr(\mathbf{y_i}|\mathbf{x_i};\theta) \tag{4.1}$$

$$= \arg\max_{\theta} \prod_i \sum_h Pr(\mathbf{y_i}, \mathbf{h}|\mathbf{x_i};\theta) \tag{4.2}$$

The expression $Pr(\mathbf{y_i}, \mathbf{h}|\mathbf{x_i})$ for a given entity pair is defined by two types of factors in the factor graph. They are *extract factors* for each mention and *mention factors* between a relation label and all the mentions. $Pr(\mathbf{y}, \mathbf{h}|\mathbf{x};\theta)$ is defined as follows:

$$Pr(\mathbf{Y} = \mathbf{y}, \mathbf{H} = \mathbf{h}|\mathbf{x};\theta) = \frac{1}{\mathcal{Z}} \prod_r f_{mention}(y^r, \mathbf{h}) \prod_i f_{extract}(h_i, x_i) \tag{4.3}$$

where $\mathcal{Z}$ is a normalization constant. The *mention factors* capture the dependency between a relation label and its mentions. Here, the *at least one* assumption that was discussed in Section 1 is modeled. It is implemented as a simple *deterministic-or* operator as given below:

$$f_{mention}(y^r, \mathbf{h}) = \begin{cases} 1 & \text{if } y^r \text{ is true } \wedge \exists i : h_i = r \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

The *extract factors* capture the local signal for each mention and consists of a bunch of lexical and syntactic features such as POS tags, dependency path between the entities,

Algorithm 1 Hoffmann et al. (2011) : Training

procedure HOFFMANN_TRAIN($\Sigma, \mathcal{E}, \mathcal{R}, \Delta$)
  for $t := 1$ to T **do** // No. of training iterations
    for $i := 1$ to N **do** // No. of entity pairs
      $\widehat{\mathbf{y_i}}, \widehat{\mathbf{h_i}}, = \arg\max_{\mathbf{y},\mathbf{h}} Pr\big(\mathbf{y_i}, \mathbf{h_i} \big| \mathbf{x_i}; \Theta\big)$
      if $\widehat{\mathbf{y_i}} \neq \mathbf{y_i}$ **then**
        $\mathbf{h_i}^* = \arg\max_{\mathbf{h}} Pr\big(\mathbf{h_i} \big| \mathbf{y_i}, \mathbf{x_i}; \Theta\big)$
        $\Theta^{new} = \Theta^{old} + \Phi(\mathbf{x_i}, \mathbf{h_i}^*) - \Phi(\mathbf{x_i}, \widehat{\mathbf{h_i}})$
  return Θ

and the like Mintz et al. (2009). They are given by the following expression:

$$f_{extract}(h_i, x_i) = exp\Big(\sum_j \theta_j(h_i, x_i)\phi_j\Big) \tag{4.5}$$

where the features ϕ_j are sensitive to the relation name assigned to extraction variable h_i, if any, and cues from the sentence x_i (Hoffmann et al., 2011).

Training algorithm

The learning algorithm is a perceptron-style parameter update scheme with two modifications: i) online learning ii) Viterbi approximation. The inference is shown to reduce to the well-known weighted edge-cover problem that can be solved exactly (via maximal weighted bipartite matching), although Hoffmann et al. (2011) provide an approximate solution. Instead of computing the maximum weighted bipartite matching, they add only those highest weighted edges incident on each of the relation nodes in the graphical model that do not violate the "at-least-one" constraint. Hence it is a greedy approximation. The procedure during the training (which involves inference as subroutines) is outlined in Algorithm 1. The input to the algorithm is as follows:

1. Σ: set of sentences
2. $\mathcal{E}$: set of entities mentioned in the sentences
3. $\mathcal{R}$: set of relation labels
4. Δ: database of facts

The output is the extraction model : Θ

4.3 Inference via Integer Linear Programming Framework

In the training algorithm described above, there are two MAP inference procedures. We use the same algorithm for training in our approach. However, we replace the inference procedures. Our contributions in this space are two-fold. Firstly, we have formulated these as ILP problems. As a result of this, the approximate inference is replaced by an exact inference procedure. Secondly, we have replaced the *deterministic-or* by a *noisy-or* which provides a soft constraint instead of the hard constraint of Hoffmann. (*"at least one"* assumption.)

Our ILP formulations

Described below are the notations and the ILP formulations for a given entity pair in our training dataset.

Some notations

- h^{jr} : The mention variable h^j (or jth sentence) using the relation value r
- s^{jr} : Score for h^j by using the value of r. Scores are computed from the *extract* factors
- y^r : relation label being r
- m : number of mentions (sentences) for the given entity pair
- R: total number of relation labels (excluding the *nil* label)

Deterministic-or

The following is the ILP formulation for the exact inference $\arg\max Pr(\mathbf{y}, \mathbf{z}|\mathbf{x_i})$ in the model based on *deterministic-or*:

$$\max_{H,Y} \left\{ \sum_{j=1}^{m} \sum_{r \in \{R,nil\}} \left[h^{jr} s^{jr} \right] \right\}$$

$$\textbf{s.t} \quad 1. \quad \sum_{r \in \{R,nil\}} h^{jr} = 1 \quad \forall j$$

$$2. \quad h^{jr} \leq y^r \quad \forall j, \forall r$$

$$3. \quad y^r \leq \sum_{j=1}^{m} h^{jr} \quad \forall r$$

$$\text{where} \quad h^{jr} \in \{0,1\}, \quad y^r \in \{0,1\}$$

The first constraint restricts a mention to take only one label. The second and third constraints impose the *at least one* assumption. This is the same formulation as Hoffmann expressed here as an ILP problem. However, posing the inference as an ILP allows us to add more constraints to it easily.

Noisy-or

As a case-study, we add the *noisy-or* soft constraint in the objective function above. The idea is to model the situation in which a fact is present in the database but is not instantiated in the text. This is a common scenario, since the facts populated in the database and the text of the corpus can come from different domains and there might not be a good match.

$$\max_{H,Y,\epsilon} \left\{ \left(\sum_{j=1}^{m} \sum_{r \in \{R,nil\}} \left[h^{jr} s^{jr} \right] \right) - \left(\sum_{r \in R} \epsilon_r \right) \right\}$$

$$\textbf{s.t} \quad 1. \quad \sum_{r \in \{R,nil\}} h^{jr} = 1 \quad \forall j$$

$$2. \quad h^{jr} \leq y^r \quad \forall j, \forall r$$

$$3. \quad y^r \leq \sum_{j=1}^{m} h^{jr} + \epsilon_r \quad \forall r$$

$$\text{where} \quad h^{jr} \in \{0,1\}, \quad y^r \in \{0,1\}, \quad \epsilon_r \in \{0,1\}$$

In the above formulation, the objective function is augmented with a soft penalty. Also, the third constraint is modified with this penalty term. We call this new term ϵ_i; It also

is a binary variable to model noise. Through this term, we encourage the *at least one* type of configuration but will not disallow a configuration that does not conform to this. Essentially, the consequence of this is to allow the case where a fact is present in the database but is not instantiated in the text.

Comparison to Related Work

Relation Extraction in the paradigm of distant supervision was introduced by Craven and Kumlien (1999). They used a biological database as the source of distant supervision to discover relations between biological entities. The progression of models for information extraction using distant supervision was presented in Section 4.1.1.

Surdeanu et al. (2012) discuss a *noisy-or* method to combine the scores of various sentence-level models to rank a relation during evaluation. In our approach, we introduce the *noisy-or* mechanism in the training phase of the algorithm.

Our work is inspired by previous works such as Roth and Yih (2004). The use of ILP for this problem facilitates easy incorporation of different constraints, and to the best of our knowledge, has not been investigated by the community.

4.4 Experiments

The experimental runs were carried out using the publicly available Stanford's distantly supervised slot-filling system[3] (Surdeanu et al., 2011) and the system provided by Hoffmann et al. (2011)[4]. The systems are implemented in Java. The system by Surdeanu et al. (2011) implements models of the following papers: Mintz et al. (2009); Riedel et al. (2010); Hoffmann et al. (2011); and Surdeanu et al. (2012). The system is an official entry in the TAC-KBP 2011 shared tasks.

4.4.1 Datasets and Evaluation

We report results on two standard datasets — Riedel and KBP — used as benchmarks by the community. A complete description of these datasets is presented in Surdeanu et al. (2012).

[3] `http://nlp.stanford.edu/software/mimlre.shtml`

[4] `http://www.cs.washington.edu/ai/raphaelh/mr/`

1. *Riedel dataset*: This dataset is constructed by aligning Freebase relations with the New York Times (NYT) corpus. The authors of this dataset are Riedel et al. (2010).

2. *KBP dataset*: This is constructed from resources distributed for the 2010 and 2011 KBP shared tasks. The knowledge base is a subset of English Wikipedia infoboxes from a 2008 snapshot. They are aligned with documents from (i) a collection provided by shared task (ii) a complete snapshot of English Wikipedia from June 2010.

Some statistics of the two datasets from Surdeanu et al. (2012) in Figure 4.3:

	# of gold relations in training	# of gold relations in testing	% of gold entity tuples with more than one label in training	% of gold entity tuples with multiple mentions in text in training	% of mentions that do not express their relation	# of relation labels
Riedel	4,700	1,950	7.5%	46.4%	up to 31%	51
KBP	183,062	3,334	2.8%	65.1%	up to 39%	41

Figure 4.3: Statistics of datasets from Surdeanu et al. (2012)

The evaluation setup and module is the same as that described in Surdeanu et al. (2012). We also use the same set of features used by the various systems in the package to ensure that the approaches are comparable. As in previous work, we report precision/recall (P/R) graphs to evaluate the various techniques. We used the publicly available *lp_solve* package[5] to solve our inference problems.

Performance of ILP

The use of ILP raises concerns about performance since it is NP-hard. In our problem, we solve a separate ILP for every entity pair. The number of variables is limited by the number of mentions for the given entity pair. Empirically, on the KBP dataset (larger of the two datasets,) Hoffmann takes around 1hr to run. Our ILP formulation takes around 8.5 hours; however, the algorithm by Surdeanu et al. (2012) (EM-based) takes around 23 hours to converge.

4.4.2 Experimental Results and Discussion

We would primarily like to highlight two settings on which we report the Precision-Recall (P/R) curves and contrast it with Hoffmann et al. (2011). Firstly, we replace the

[5] `http://lpsolve.sourceforge.net/5.5/`

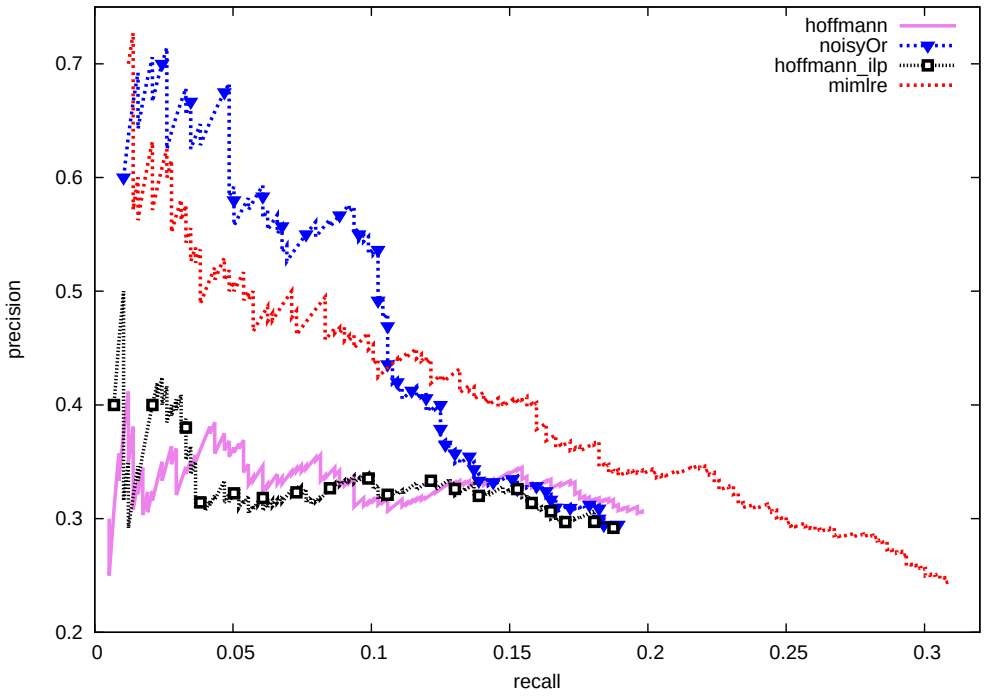


Figure 4.4: Results : KBP dataset

approximate inference in that work with our ILP-based exact inference; we call this setting the `hoffmann-ilp`. Secondly, we replace the *deterministic-or* in the model with a *noisy-or*, and call this setting the `noisy-or`. We further compare our approach with Surdeanu et al. (2012) (`mimlre`). The P/R curves for the various techniques on the two datasets are shown in Figures 4.4 and 4.5. We also report the highest F_1 point in the P/R curve for both the datasets in Tables 4.1 and 4.2.

Discussion

We would like to discuss the results in the above two scenarios.

1. **Performance of `hoffmann-ilp`**

   On the KBP dataset, we observe that `hoffmann-ilp` has higher precision in the range of 0.05 to 0.1 at lower recall (0 to 0.04). In other parts of the curve, it is very close to the baseline (although hoffmann's algorithm is slightly better). In Table 4.1, we notice that the recall of `hoffmann-ilp` is lower in comparison to

	Precision	Recall	F1
`hoffmann`	30.645	19.791	24.050
`mimlre`	28.061	28.645	28.350
`noisy-or`	29.700	18.923	23.117
`hoffmann-ilp`	29.301	18.923	22.996

Table 4.1: Highest F1 point in P/R curve: KBP Dataset

	Precision	Recall	F1
`hoffmann`	32.054	24.049	27.480
`mimlre`	28.061	28.645	28.350
`noisy-or`	31.700	18.139	23.075
`Hoffmann-ilp`	36.701	12.692	18.862

Table 4.2: Highest F1 point in P/R curve: Riedel Dataset

hoffmann's algorithm.

On the Riedel dataset, we observe that `hoffmann-ilp` has better precision (0.15 to 0.2) than `mimlre` within recall of 0.1. At recall > 0.1, precision drops drastically. This is because `hoffmann-ilp` predicts significantly more nil labels. However, nil labels are not part of the label-set in the P/R curves reported in the community. In Table 4.2, we see that `hoffmann-ilp` has higher precision (0.04) compared to Hoffmann's algorithm.

2. **Performance of `noisy-or`**

In Figure 4.4 we see that there is a big jump in the precision (around 0.4) of `noisy-or` compared to Hoffmann's model in most parts of the curve on the KBP dataset. However, in Figure 4.5 (Riedel dataset), we do not see such a trend. However, we do perform better than `mimlre` by Surdeanu et al. (2012) (precision > 0.15 for recall < 0.15).

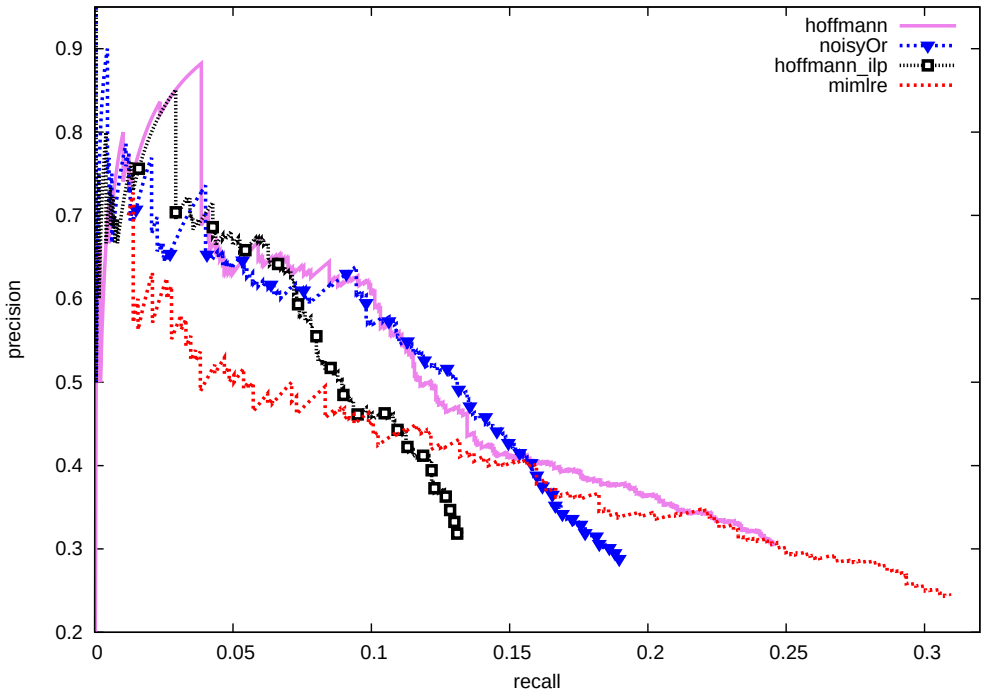


Figure 4.5: Results : Riedel dataset

On both datasets, `noisy-or` has higher precision than `mimlre`, as seen from Tables 4.1 and 4.2. However, the recall reduces. More investigation in this direction is part of future work.

4.5 Chapter Summary

In this chapter we have described an important addition to Hoffmann's model through the use of the *noisy-or* soft constraint to further relax *the at least one* assumption. Since we posed the inference procedures in Hoffmann using ILP, we could easily add this constraint during the training and inference.

Empirically, we showed that the resulting P/R curves have a significant performance boost over Hoffmann's algorithm as a result of this newly added constraint. Although our system has a lower recall when compared to `mimlre` (Surdeanu et al., 2012), it performs competitively with respect to the precision at low recall.

Figure 4.6 highlights the part presented in this chapter and the accompanying pub-

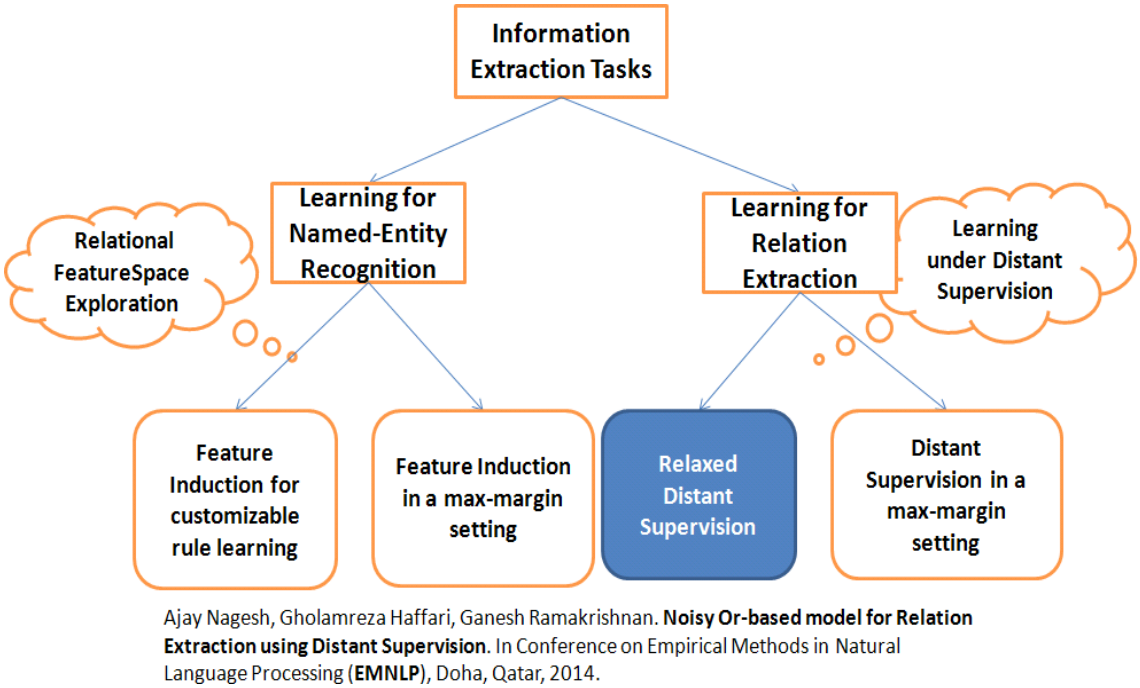


Figure 4.6: Thesis Organization: Chapter 4

lication that overlaps with the material presented.

Chapter 5

Distant Supervision in a Max-margin Setting

In this Chapter, we present a large-margin method to learn parameters of latent variable models for a wide range of non-linear multivariate performance measures such as F_β. Our method can be applied to learning graphical models that incorporate inter-dependencies among the output variables directly or indirectly, through hidden variables.

5.1 Latent Variable Structure Prediction Tasks

Rich models with latent variables are popular in many problems in natural language processing. For instance, in Information Extraction, one needs to predict the relation labels $\mathbf{y}$ that an entity-pair $\mathbf{x}$ can take based on the hidden relation mentions $\mathbf{h}$, that is, the relation labels for occurrences of the entity-pair in a given corpus. However, these models are often trained by optimizing performance measures (such as conditional log-likelihood or error rate) that are not directly related to the task-specific non-linear performance measure, for example, the F_1-score.

Similarly, state-of-the-art *Machine Translation* models describe the translation process from the source sentence $\mathbf{x}$ to the target sentence $\mathbf{y}$ through the latent linguistic structure $\mathbf{h}$, for example, parse tree or the alignment.

A typical approach is to populate and fix the latent alignments in the early stages of the pipeline, and then to learn parameters at later stages by optimizing for the translation performance. However, better models may be trained by optimizing the task-specific performance measure while allowing latent variables to adapt their values accordingly.

5.1.1 Related Work

Many machine learning problems have a very rich structure associated with the features in the input as well as output spaces. Modeling this rich structure is facilitated by graphical models such as Bayesian networks and Markov Random Fields. There is a wide variety of techniques to learn the parameters of the model. The broad categories of training algorithms are generative and discriminative. A typical generative approach maximizes the likelihood of the training data modeling the joint probability $Pr(\mathbf{x}, \mathbf{y})$ of input features and output labels. A discriminative model directly models the dependency of $\mathbf{y}$ or $\mathbf{x}$ either by modeling $Pr(\mathbf{y}|\mathbf{x})$ or by learning a discriminant function in the case of *Support Vector Machines (SVMs).*

In general, discriminative approaches are shown to perform much better when compared to generative approaches, especially on large datasets (Ng and Jordan, 2002). Large-margin methods, in particular, SVMs, are a powerful class of learners that are used effectively in large structure prediction tasks (Taskar et al., 2003). SVMs are a highly flexible framework, in which a number of modeling complexities can be easily incorporated. For instance, models that have hidden variables can be cast in the SVM formulation (Yu and Joachims, 2009). The SVM framework also provides techniques to learn models that optimize non-decomposable performance measures such as the F_1 score which is computed on the entire data sample.

As stated earlier, large-margin methods have been shown to be a compelling approach to learn rich models that detail the inter-dependencies among the output variables. This is achieved by optimizing loss functions that are either *decomposable* over the *training instances* (Taskar et al., 2003; Tsochantaridis et al., 2004) or *non-decomposable loss functions* (Ranjbar et al., 2013; Tarlow and Zemel, 2012; Rosenfeld et al., 2014; Keshet, 2014). Large-margin methods have also been shown to be powerful when applied to latent variable models — when optimizing for decomposable loss functions (Wang and Mori, 2011; Felzenszwalb et al., 2010; Yu and Joachims, 2009).

5.1.2 Our Contributions

Our large-margin method learns latent variable models by optimizing non-decomposable loss functions. It interleaves the Concave-Convex Procedure (CCCP) (Yuille and Rangarajan, 2003) to populate latent variables with dual decomposition (Komodakis et al.,

2011; Rush and Collins, 2012). The latter factorizes the hard optimization problem (encountered in learning) into smaller independent sub-problems over the training instances. We present linear programming and local search methods for effective optimization of the sub-problems encountered in the dual decomposition. Our local search algorithm leads to a speed up of 7,000 times compared to the exhaustive search used in the literature (Joachims, 2005; Ranjbar et al., 2012).

Our work is the first of its kind in the use of max-margin training in distant supervision of relation extraction models. We demonstrate the effectiveness of our proposed method in comparison with two strong baseline systems that optimize for the error rate and conditional likelihood — including a state-of-the-art system by Hoffmann et al. (2011). On several data conditions, we show that our method outperforms the baseline and results in up to 8.5% improvement in the F_1-score.

In addition, our approach can be applied more generally to a class of loss functions, the maximum value of which can be computed efficiently. As an example, our method can be applied to graphical models that incorporate inter-dependencies among the output variables either directly, or indirectly through hidden variables.

5.2 Max-margin Formulation

5.2.1 Distant Supervision as a Large-margin Problem

Our framework is motivated by distant supervision for learning relation extraction models (Mintz et al., 2009). The goal is to learn relation extraction models by aligning facts in a database to sentences in a large unlabeled corpus. Since the individual sentences are not hand labeled, the facts in the database act as "weak" or "distant" labels; hence, the learning scenario is termed as distantly supervised.

Prior work casts this problem as a multi-instance multi-label learning problem (Hoffmann et al., 2011; Surdeanu et al., 2012). It is multi-instance, since for a given entity-pair, only the label of the bag of sentences containing both entities (also known as *mentions*) is given. It is multi-label since a bag of mentions can have multiple labels. The inter-dependencies between relation labels and (hidden) mention labels are modeled by a Markov Random Field (Figure 5.1) (Hoffmann et al., 2011). The learning algorithms used in the literature for this problem optimize the (conditional) likelihood, but the evaluation

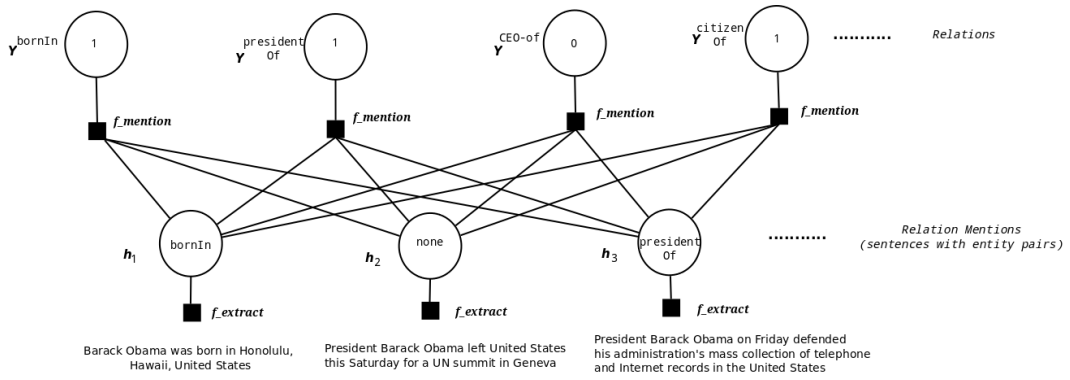


Figure 5.1: Graphical model instantiated for an entity-pair

measure is commonly the F-score.

Formally, the training data is $\mathcal{D} := \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ where $\mathbf{x}_i \in \mathcal{X}$ is the entity-pair, $\mathbf{y}_i \in \mathcal{Y}$ denotes the relation labels, and $\mathbf{h}_i \in \mathcal{H}$ denotes the *hidden* mention labels. The possible relation labels for the entity-pair are observed from a given knowledge base. If there are L candidate relation labels in the knowledge base, then $\mathbf{y}_i \in \{0,1\}^L$, (for example, $y_{i,\ell}$ is 1 if the relation ℓ is licensed by the knowledge-base for the entity-pair) and $\mathbf{h}_i \in \{1, .., L, nil\}^{|\mathbf{x}_i|}$ (that is, each mention realizes one of the relation labels or *nil*.)

Notation

In the rest of this chapter, we denote the collection of all entity-pairs $\{\mathbf{x}_i\}_{i=1}^N$ by $\mathbf{X} \in \boldsymbol{\mathcal{X}} := \mathcal{X} \times .. \times \mathcal{X}$, the collection of mention relations $\{\mathbf{h}_i\}_{i=1}^N$ by $\mathbf{H} \in \boldsymbol{\mathcal{H}} := \mathcal{H} \times .. \times \mathcal{H}$, and the collection of relation labels $\{\mathbf{y}_i\}_{i=1}^N$ by $\mathbf{Y} \in \boldsymbol{\mathcal{Y}} := \mathcal{Y} \times .. \times \mathcal{Y}$.

The aim is to learn a parameter vector $\mathbf{w} \in \mathbb{R}^d$, by which the relation labels for a new entity-pair $\mathbf{x}$ can be predicted

$$f_{\mathbf{w}}(\mathbf{x}) := \arg\max_{\mathbf{y}} \max_{\mathbf{h}} \mathbf{w} \cdot \Phi(\mathbf{x}, \mathbf{h}, \mathbf{y}) \tag{5.1}$$

where $\Phi \in \mathbb{R}^d$ is a feature vector defined according to the Markov Random Field, modeling the inter-dependencies between $\mathbf{x}$ and $\mathbf{y}$ through $\mathbf{h}$ (Figure 5.1). In training, we would like to minimize the loss function Δ by which the model will be assessed at test time. For the relation extraction task, the loss can be considered to be the negative of the F_β score:

$$F_\beta = \frac{1}{\frac{\beta}{\text{Precision}} + \frac{1-\beta}{\text{Recall}}} \tag{5.2}$$

where $\beta = 0.5$ results in optimizing against F_1-score. Our proposed learning method optimizes those loss functions Δ that cannot be decomposed over individual training

instances. For example, F_β depends non-linearly on Precision and Recall which in turn requires the predictions for *all* the entity-pairs in the training set; hence it cannot be decomposed over individual training instances.

5.2.2 Introduction to Structured Prediction Learning

The goal of our learning problem is to find $\mathbf{w} \in R^d$, which minimizes the expected loss — also known as *risk* — over a new sample $\mathcal{D}'$ of size N':

$$R^{\Delta}_{f_\mathbf{w}} := \int \Delta\Big(\big(f_\mathbf{w}(\mathbf{x}'_1), .., f_\mathbf{w}(\mathbf{x}'_{N'})\big), \big(\mathbf{y}'_1, .., \mathbf{y}'_{N'}\big)\Big) dPr(\mathcal{D}') \tag{5.3}$$

Generally, the loss function Δ cannot be decomposed into a linear combination of a loss function δ over individual training samples. However, most discriminative large-margin learning algorithms assume for simplicity that the loss function is decomposable and the samples are i.i.d. (independent and identically distributed), which simplifies the sample risk $R^{\Delta}_{f_\mathbf{w}}$ as:

$$R^{\delta}_{f_\mathbf{w}} := \int \delta(f_\mathbf{w}(\mathbf{x}'), \mathbf{y}') dPr(\mathbf{x}', \mathbf{y}') \tag{5.4}$$

Often learning algorithms make use of empirical risk as an approximation of sample risk:

$$\hat{R}^{\delta}_{f_\mathbf{w}} := \frac{1}{N} \sum_{i=1}^{N} \delta(f_\mathbf{w}(\mathbf{x}_i), \mathbf{y}_i) \tag{5.5}$$

For non-decomposable loss functions such as F_β, Δ cannot be expressed in terms of instance-specific loss function δ to construct the empirical risk of the kind in Equation (5.5). Instead, we need to optimize the empirical risk that is constructed based on the sample loss:

$$\hat{R}^{\Delta}_{f_\mathbf{w}} := \Delta\Big(\big(f_\mathbf{w}(\mathbf{x}_1), .., f_\mathbf{w}(\mathbf{x}_N)\big), \big(\mathbf{y}_1, .., \mathbf{y}_N\big)\Big) \tag{5.6}$$

or equivalently

$$\hat{R}^{\Delta}_{f_\mathbf{w}} := \Delta(f_\mathbf{w}(\mathbf{X}), \mathbf{Y}) \tag{5.7}$$

where $f_\mathbf{w}(\mathbf{X}) := (f_\mathbf{w}(\mathbf{x}_1), .., f_\mathbf{w}(\mathbf{x}_N))$.

Having defined the empirical risk in Eq (5.7), we formulate the learning problem as a structured prediction problem. Instead of learning a mapping function $f_\mathbf{w} : \mathcal{X} \to \mathcal{Y}$ from an individual instance $\mathbf{x} \in \mathcal{X}$ to its label $\mathbf{y} \in \mathcal{Y}$, we learn a mapping function $\mathbf{f} : \boldsymbol{\mathcal{X}} \to \boldsymbol{\mathcal{Y}}$ from all instances $\mathbf{X} \in \boldsymbol{\mathcal{X}}$ to their labels $\mathbf{Y} \in \boldsymbol{\mathcal{Y}}$. We then define the best labeling using a linear discriminant function:

$$\mathbf{f}(\mathbf{X}) := \arg \max_{\mathbf{Y}' \in \boldsymbol{\mathcal{Y}}} \max_{\mathbf{H}' \in \boldsymbol{\mathcal{H}}} \Big\{ \mathbf{w} \cdot \Psi(\mathbf{X}, \mathbf{H}', \mathbf{Y}') \Big\} \tag{5.8}$$

where $\Psi(\mathbf{X}, \mathbf{H}', \mathbf{Y}') := \sum_{i=1}^{N} \Phi(\mathbf{x}_i, \mathbf{h}'_i, \mathbf{y}'_i)$. Based on the margin re-scaling formulation of structured prediction problems (Tsochantaridis et al., 2004), the training objective can be written as the following unconstrained optimization problem:

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||_2^2 + C \max_{\mathbf{Y}'} \Big\{ \max_{\mathbf{H}'} \mathbf{w} \cdot \Psi(\mathbf{X}, \mathbf{H}', \mathbf{Y}') - \max_{\mathbf{H}'} \mathbf{w} \cdot \Psi(\mathbf{X}, \mathbf{H}', \mathbf{Y}) + \Delta(\mathbf{Y}', \mathbf{Y}) \Big\} \tag{5.9}$$

which is similar to the training objective for the latent SVMs (Yu and Joachims, 2009). The difference is that the instance-dependent loss function δ is replaced by the sample loss function Δ. To learn $\mathbf{w}$ by optimizing the objective function mentioned above is challenging, and is the subject of the next section.

5.3 Optimizing Multi-variate Performance Measures

In this section we present our method to learn latent SVMs with non-decomposable loss functions. Our training objective is Equation (5.9), which can be equivalently expressed as:

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||_2^2 + C \max_{\mathbf{y}'_1, .., \mathbf{y}'_N} \Big\{ \Delta\Big((\mathbf{y}_1, .., \mathbf{y}_N), (\mathbf{y}'_1, .., \mathbf{y}'_N) \Big) + \sum_{i=1}^{N} \max_{\mathbf{h}} \mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{h}, \mathbf{y}'_i) - \sum_{i=1}^{N} \max_{\mathbf{h}} \mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{h}, \mathbf{y}_i) \Big\} \tag{5.10}$$

The training objective is non-convex, since it is the difference of two convex functions. In this section we make use of the CCCP to populate the hidden variables (Yu and Joachims, 2009; Yuille and Rangarajan, 2003), and interleave it with dual decomposition (DD) to solve the resulting intermediate loss-augmented inference problems (Ranjbar et al., 2012; Rush and Collins, 2012; Komodakis et al., 2011).

5.3.1 Concave-Convex Procedure (CCCP)

The CCCP (Yuille and Rangarajan, 2003) offers a general iterative method to optimize those non-convex objective functions that can be written as the difference of two convex functions $g_1(\mathbf{w}) - g_2(\mathbf{w})$. The idea is to iteratively lowerbound g_2 with a linear function $g_2(\mathbf{w}^{(t)}) + \mathbf{v} \cdot (\mathbf{w} - \mathbf{w}^{(t)})$, and take the following step to update $\mathbf{w}$:

$$\mathbf{w}^{t+1} := \arg\min_{\mathbf{w}} \big\{ g_1(\mathbf{w}) - \mathbf{w} \cdot \mathbf{v}^t \big\} \tag{5.11}$$

Algorithm 2 The Training Algorithm (Optimizing Equation 5.10)

procedure OPT-LATENTSVM($\mathbf{X}, \mathbf{Y}$)
  Initialize $\mathbf{w}^{(0)}$ and set $t = 0$
  repeat
    for $i := 1$ to N **do**
      $\mathbf{h}_i^* := \arg\max_{\mathbf{h}} \mathbf{w}^{(t)} \cdot \Phi(\mathbf{x}_i, \mathbf{h}, \mathbf{y}_i)$
    // Optimizing Equation 5.12
    $\mathbf{w}^{(t+1)} := \text{optSVM}(\mathbf{X}, \mathbf{H}^*, \mathbf{Y})$
    $t := t + 1$
  until some stopping condition is met
  return $\mathbf{w}^{(t)}$

In our case, the training objective in Equation (5.10) is the difference of two convex functions where the second function g_2 is $C \sum_{i=1}^{N} \max_{\mathbf{h}} \left\{\mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{h}, \mathbf{y}_i)\right\}$. The upperbound of $g_1(\mathbf{w}) - g_2(\mathbf{w})$ involves populating the hidden variables by:

$$\mathbf{h}_i^* := \arg\max_{\mathbf{h}} \left\{\mathbf{w}^{(t)} \cdot \Phi(\mathbf{x}_i, \mathbf{h}, \mathbf{y}_i)\right\}.$$

Therefore, in each iteration of our CCCP-based algorithm we need to optimize Eq (5.11) — reminiscent of the standard structural SVM without latent variables:

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||_2^2 + C \max_{\tilde{\mathbf{y}}_1,..,\tilde{\mathbf{y}}_N} \left\{ \Delta\Big((\mathbf{y}_1,..,\mathbf{y}_N), (\tilde{\mathbf{y}}_1,..,\tilde{\mathbf{y}}_N)\Big) + \sum_{i=1}^{N} \max_{\mathbf{h}} \mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{h}, \tilde{\mathbf{y}}_i) - \sum_{i=1}^{N} \mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{h}_i^*, \mathbf{y}_i) \right\} \tag{5.12}$$

The objective function mentioned above can be optimized using the standard cutting-plane algorithms for structural SVM (Tsochantaridis et al., 2004; Joachims, 2005). The cutting-plane algorithm in turn needs to solve the *loss-augmented inference*, which is the subject of the next sub-section. The CCCP-based training algorithm is summarized in Algorithm 2.

5.3.2 Loss-Augmented Inference

To be able to optimize the training objective in Equation (5.12), which is encountered in each iteration of Algorithm 2, we need to solve (the so-called) loss-augmented inference:

$$\max_{\mathbf{y}'_1,..,\mathbf{y}'_N} \quad \Delta\Big((\mathbf{y}_1,..,\mathbf{y}_N),(\mathbf{y}'_1,..,\mathbf{y}'_N)\Big) + \sum_{i=1}^{N} \max_{\mathbf{h}} \mathbf{w}\cdot\Phi(\mathbf{x}_i,\mathbf{h},\mathbf{y}'_i) \tag{5.13}$$

We make use of the dual decomposition (DD) technique to decouple the two terms of the above objective function, and efficiently find an approximate solution. DD is shown to be an effective technique for loss-augmented inference in structured prediction models *without* hidden variables (Ranjbar et al., 2012).

To apply DD to the loss-augmented inference in Equation (5.13), let us rewrite it as a constrained optimization problem:

$$\begin{aligned}
&\max_{\mathbf{y}'_1,\ldots,\mathbf{y}'_N,\mathbf{y}''_1,\ldots,\mathbf{y}''_N} \Delta\Big((\mathbf{y}_1,\ldots,\mathbf{y}_N),(\mathbf{y}'_1,\ldots,\mathbf{y}'_N)\Big) \\
&\qquad + \sum_{i=1}^{N} \max_{\mathbf{h}} \mathbf{w}\cdot\Phi(\mathbf{x}_i,\mathbf{h},\mathbf{y}_i{}'') \\
&\text{subject to} \\
&\forall i \in \{1,\ldots,N\}, \forall \ell \in \{1,\ldots,L\}, \qquad y'_{i,\ell} = y''_{i,\ell}
\end{aligned}$$

Introduction of the new variables $(\mathbf{y}''_1,..,\mathbf{y}''_N)$ decouples the two terms in the objective function, and leads to an effective optimization algorithm. After forming the Lagrangian, the dual objective function is derived as:

$$\begin{aligned}
L(\mathbf{\Lambda}) := &\max_{\mathbf{Y}'} \Delta(\mathbf{Y},\mathbf{Y}') + \sum_i \sum_\ell \lambda_i(\ell) y'_{i,\ell} + \\
&\max_{\mathbf{Y}''} \sum_{i=1}^{N} \max_{\mathbf{h}} \mathbf{w}\cdot\Phi(\mathbf{x}_i,\mathbf{h},\mathbf{y}_i{}'') - \sum_i \sum_\ell \lambda_i(\ell) y''_{i,\ell}
\end{aligned}$$

where $\mathbf{\Lambda} := (\boldsymbol{\lambda}_1,..,\boldsymbol{\lambda}_N)$, and $\boldsymbol{\lambda}_i$ is a vector of Lagrange multipliers for L binary variables, each of which represents a relation label. The two optimization problems involved in the dual $L(\mathbf{\Lambda})$ are independent and can be solved separately. The dual is an upperbound on the loss-augmented objective function for any value of $\mathbf{\Lambda}$; therefore, we can find the tightest upperbound as an approximate solution:

$$\min_{\mathbf{\Lambda}} L(\mathbf{\Lambda})$$

Algorithm 3 Loss-Augmented Inference

procedure OPT-LOSSAUG($\mathbf{w}, \mathbf{X}, \mathbf{Y}$)
  Initialize $\mathbf{\Lambda}^{(0)}$ and set $t = 0$
  repeat
    $\mathbf{Y}_*^{'} :=$ opt-LossLag($\mathbf{\Lambda}, \mathbf{Y}$) // Eq. (5.14)
    $\mathbf{Y}_*^{''} :=$ opt-ModelLag($\mathbf{\Lambda}, \mathbf{X}$) // Eq. (5.15)
    if $\mathbf{Y}_*^{'} = \mathbf{Y}_*^{''}$ **then**
      return $\mathbf{Y}_*^{'}$
    for $i := 1$ to N **do**
      for $\ell := 1$ to L **do**
        $\lambda_i^{(t+1)}(\ell) := \lambda_i^{(t)}(\ell) - \eta^{(t)}(y'_{i,\ell} - y''_{i,\ell})$
  until some stopping condition is met
  return $\mathbf{Y}_*^{'}$

The dual is non-differentiable at those points $\mathbf{\Lambda}$ where either of the two optimization problems has multiple optima. Therefore, it is optimized using the subgradient descent method:

$$\Lambda^{(t)} := \Lambda^{(t-1)} - \eta^{(t)}(\mathbf{Y}_*^{'} - \mathbf{Y}_*^{''})$$

where $\eta^{(t)} = \frac{1}{\sqrt{t}}$ is the step size[1], and

$$\mathbf{Y}_*^{'} := \arg\max_{\mathbf{Y}'} \Delta(\mathbf{Y}, \mathbf{Y}') + \sum_i \sum_\ell \lambda_i^{(t-1)}(\ell) y'_{i,\ell} \tag{5.14}$$

$$\mathbf{Y}_*^{''} := \arg\max_{\mathbf{Y}''} \sum_{i=1}^{N} \max_{\mathbf{h}} \mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{h}, \mathbf{y}_i{}'') - \sum_i \sum_\ell \lambda_i^{(t-1)}(\ell) y''_{i,\ell} \tag{5.15}$$

The DD algorithm to compute the loss-augmented inference is outlined in Algorithm 3. The challenge lies in effectively solving the two optimization problems mentioned above, which is the subject of the following section.

[1]Other (non-increasing) functions of the iteration number t are also plausible, as far as they satisfy the following conditions (Komodakis et al., 2011) that are needed to guarantee the convergence to the optimal solution in the subgradient descent method: $\eta^{(t)} \geq 0, \lim_{t\to\infty} \eta^{(t)} = 0, \sum_{t=1}^{\infty} \eta^{(t)} = \infty$

Algorithm 4 Finding $\mathbf{Y}'_*$: Local Search

procedure OPT-LOSSLAG($\mathbf{\Lambda}, \mathbf{Y}$)
  $(idx_1^n \dots idx_{\#neg}^n) \leftarrow$ Sort $\downarrow (\lambda_i(\ell))$ // FPs
  $(idx_1^n \dots idx_{\#pos}^n) \leftarrow$ Sort $\uparrow (\lambda_i(\ell))$ // FNs
  Initialize (fp, fn) on the grid
  repeat
    for $((fp', fn') \in$ `Neigbours`(fp, fn) **do**
      $loss_{(fp',fn')} = \Delta(fp', fn') + \mathbf{\Lambda}_{sorted}$[3]
    $loss_{(fp'',fn'')} = \arg\max_{(fp',fn')} loss_{(fp',fn')}$
    if $loss_{(fp,fn)} > loss_{(fp'',fn'')}$ **then**
      break
    else
      $(fp, fn) = (fp'', fn'')$
  until $loss_{(fp,fn)} \leq loss_{(fp'',fn'')}$
return { $\mathbf{Y}'$ corresponding to (fp, fn) }

5.3.3 Effective Optimization of the Dual

The two optimization problems that are involved in the dual are hard in general. More specifically, the optimization of the affine-augmented model score (in Equation 5.15) is as difficult as the MAP inference in the underlying graphical model, which can be challenging for loopy graphs. For the graphical model underlying distant supervision of relation extraction (Fig 5.1), we formulate the inference as an ILP (integer linear program) as explained in Chapter 4. Furthermore, we relax the ILP to LP to speed up the inference, at the expense of trading exact solutions with approximate solutions[2].

Likewise, the optimization of the affine-augmented multivariate loss (in Equation 5.14) is difficult. This is because we have to search over the entire space of $\mathbf{Y}' \in \mathcal{Y}$, which is exponentially large $\mathcal{O}(2^{N*L})$. However, if the loss term Δ can be expressed in terms of some aggregate statistics over $\mathbf{Y}'$, such as false positives (FPs) and false negatives (FNs),

[2]We observed in our experiments that relaxing the ILP to LP does not hurt the performance, but speeds up the inference significantly.

[3]For a given (fp, fn), we set $\mathbf{y}'$ by picking the sorted unary terms that maximize the score according to $\mathbf{y}$.

the optimization can be performed efficiently. This is due the fact that the number of FPs can range from zero to the size of the negative labels, and the number of FNs can range from zero to the number of positive labels. Therefore, the loss term can take $\mathcal{O}(N^2L^2)$ different values which can be represented on a two-dimensional grid. After fixing FPs and FNs to a grid point, $\mathbf{\Lambda} \cdot \mathbf{Y}'$ is maximized with respect to $\mathbf{Y}'$. The grid point that has the best value for $\Delta(\mathbf{Y}, \mathbf{Y}') + \mathbf{\Lambda} \cdot \mathbf{Y}'$ will then provide the optimal solution for the Equation (5.14).

An exhaustive search in the space of all possible grid points becomes inefficient as soon as the grid becomes large. Therefore, we have to adapt the techniques proposed in previous work (Ranjbar et al., 2012; Joachims, 2005). We propose a simple but effective *local search* strategy for this purpose. The procedure is outlined in Algorithm 4. We start with a random grid point, and move to the best neighbor. We keep *hill climbing* until there is no neighbor better than the current point. We define the neighborhood by a set of exponentially spaced points around the current point, to improve the exploration of the search space. We present some analysis on the benefits of using this search strategy vis-à-vis the exhaustive search in the Experiments section.

5.4 Experiments

5.4.1 Experimental Setup

Dataset

We use the challenging benchmark dataset created by Riedel et al. (2010) for distant supervision of relation extraction models. It is created by aligning relations from *Freebase*[4] with the sentences in the *New York Times* corpus (Sandhaus, 2008). The labels for the data points come from the Freebase database; however, Freebase is incomplete (Ritter et al., 2013). So a data point is labeled *nil* when either no relation exists or the relation is absent in Freebase. To avoid this ambiguity, we train and evaluate the baseline and our algorithms on a subset of this dataset, which consists of only non-nil relation labeled data points (termed as *positive dataset*). For the sake of completeness, we report the accuracies of the various approaches on the entire evaluation dataset.

[4] `www.freebase.com`

Systems and Baseline

Hoffmann et al. (2011) describe a state-of-the-art approach for this task. They use a perceptron-style parameter update scheme adapted to handle latent variables; their training objective is the conditional likelihood. Out of the two implementations of this algorithm, we use the better[5] of these two[6] as our baseline (denoted by `Hoffmann`). For a fair comparison, the training dataset and the set of features defined over it are common to all the experiments.

We discuss the results of two of our approaches. One is the LatentSVM max-margin formulation with the simple decomposable Hamming loss function, which minimizes the error rate (denoted by `MM-hamming`). The other is the LatentSVM max-margin formulation with the non-decomposable loss function, which minimizes the negative of F_β score (denoted by `MM-F-loss`)[7].

Evaluation Measure

The performance measure is F_β, which can be expressed in terms of FP and FN as:

$$F_\beta = \frac{N_p - FN}{\beta(FP - FN) + N_p}$$

where β is the weight assigned to precision (and $1 - \beta$ to recall). FP, FN and N_p are defined as :

$$\begin{aligned} FP &= \sum_i \sum_\ell y'_{i,\ell}(1 - y_{i,l}) \\ FN &= \sum_i \sum_\ell y_{i,\ell}(1 - y'_{i,l}) \\ N_p &= \sum_i \sum_\ell y_{i,\ell} \end{aligned}$$

We use $1 - F_\beta$ as the expression for the multivariate loss.

5.4.2 Training on Sub-samples of Data

We performed a number of experiments using different randomized subsets of the Riedel dataset (10% of the positive dataset) to train the max-margin approaches. This was done

[5]It is not clear why the performance of the two implementations is different.

[6]`nlp.stanford.edu/software/mimlre.shtml`

[7]We use a combination of F1 loss and Hamming loss, since using only F1 loss overfits the training dataset, as observed from the experiments.

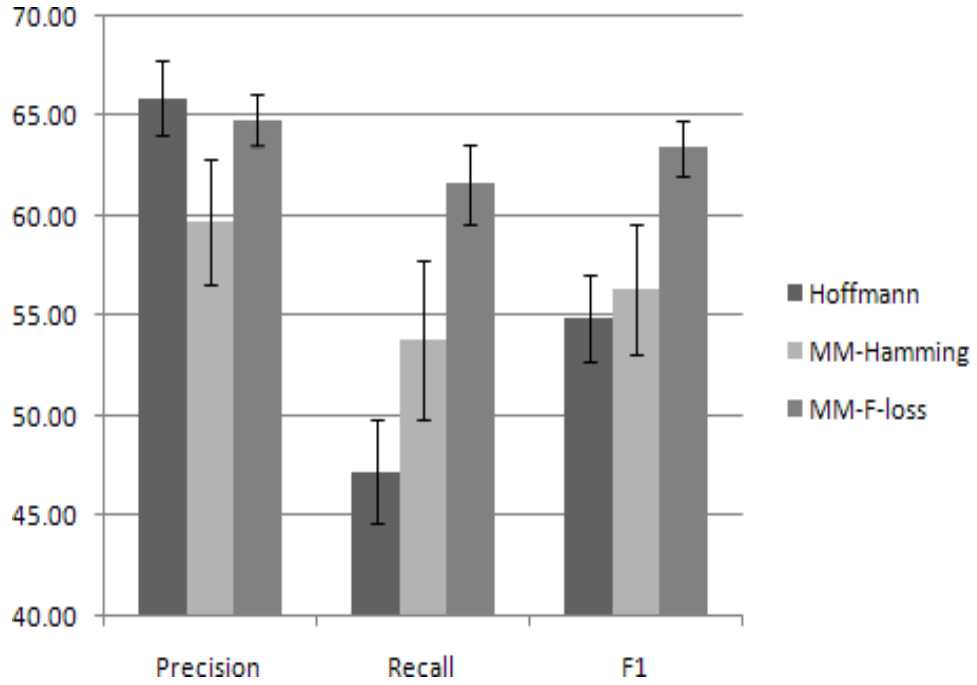


Figure 5.2: Experiments on 10% Riedel datasets.

	Precision	*Recall*	F_1
Hoffmann	65.93	47.22	54.91
MM-Hamming	59.74	53.81	56.32
MM-F-loss	64.81	61.63	63.44

Table 5.1: Average results on 10% Riedel datasets.

in order to determine empirically a good set of parameters for training. We also compare the results of the approaches with `Hoffmann` trained on the same sub-samples.

Comparison with the Baseline

We report the average over 15 subsets of the dataset with a 90% confidence interval (using student-t distribution). The results of these experiments are shown in Figure 5.2 and Table 5.1. We observe that both `MM-Hamming` and `MM-F-loss` have a higher F_1-score compared to the baseline. There is a significant improvement in the F_1-score to the tune of 8.52% for the multivariate performance measure over `Hoffmann`. There is also an improvement in the F_1-score of 7.12%, compared to `MM-Hamming`. This highlights

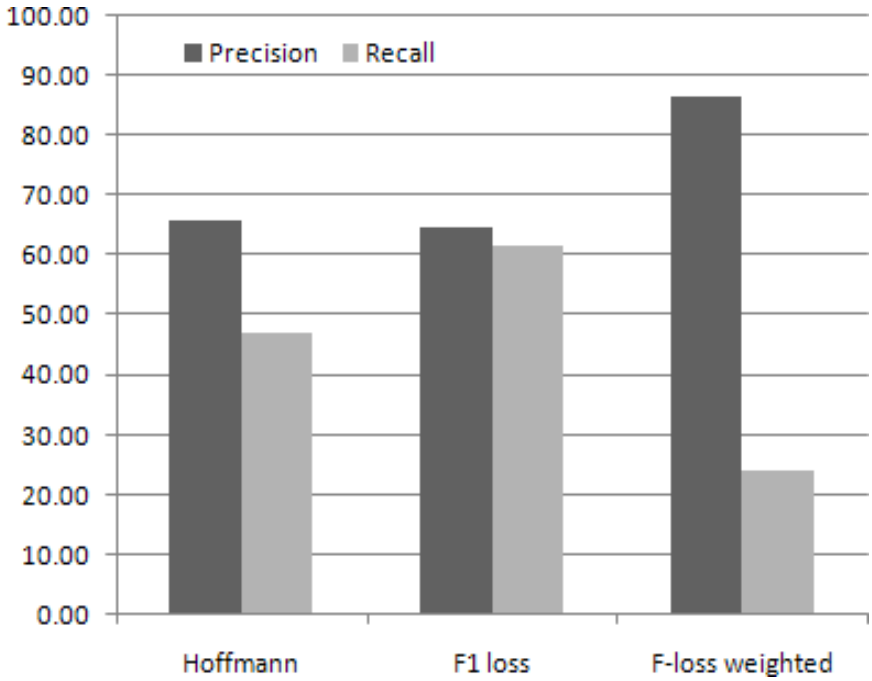


Figure 5.3: Weighting of Precision and Recall

the importance of using non-linear loss functions, when compared to using simple loss functions such as error rate during training.

However, `Hoffmann` has a marginally higher precision of about 1.13%. We noticed that this was due to over-fitting of the model on the data, since the performance on the training datasets was very high. Another interesting observation of `MM-F-loss` is that it is fairly balanced with respect to both Precision and Recall, which the other approaches do not exhibit.

Tuning toward Precision/Recall

Often, we come across situations where either precision or recall is important for a given application. This is modeled by the notion of F_β (van Rijsbergen, 1979). One of the main advantages of using a non-decomposable loss function such as F_β is the ability to vary the learning algorithm to factor in such situations. For instance, we can tune the objective to favor precision more than recall by "up-weighting" precision in the F_β-score.

As an illustration, in the previous case, we observed that `MM-F-loss` has a marginally poorer precision compared to `Hoffmann`. Suppose we increase the weight of precision, $\beta = 0.833$, we observe a dramatic increase in precision from 65.83% to 86.59%. As expected, due to the precision-recall trade-off, we observe a decrease in recall. The results

	avg. time per iter.	F_1
Local Search	0.09s	58.322
Exhaustive Search	630s	58.395

Table 5.2: Local vs. Exhaustive Search.

are shown in Figure 5.3.

Local versus Exhaustive Grid Search

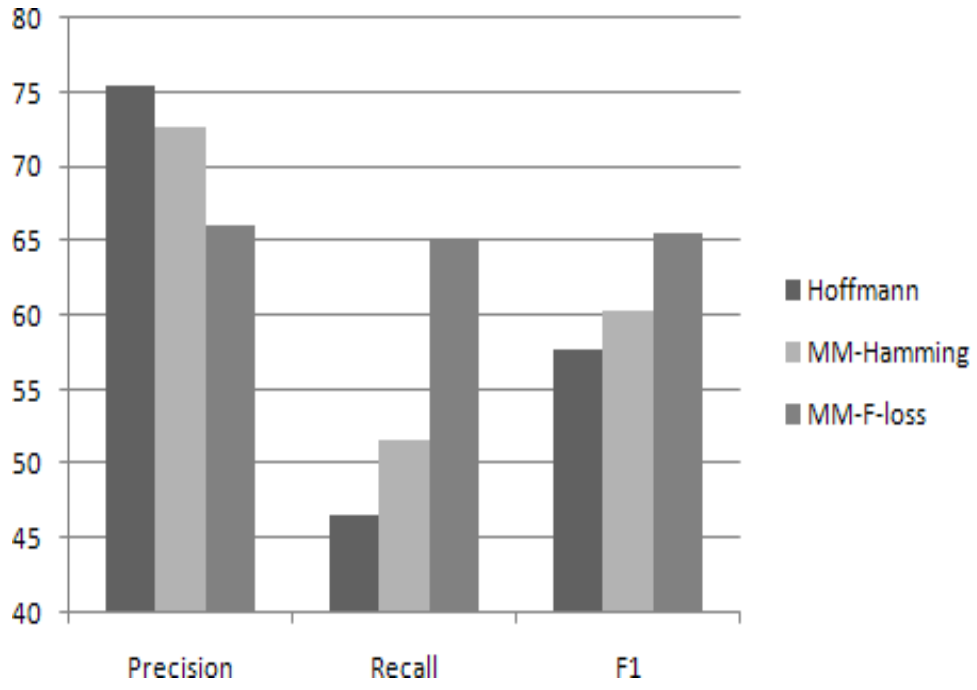


Figure 5.4: Overall accuracies Riedel dataset

As we described in Section 5.3.3, we devise a simple yet efficient local search strategy to search the space of (FP, FN) grid-points. This enables a speed up of three orders of magnitude in solving the dual-optimization problem. In Table 5.2, we compare the average time per iteration and the F_1-score when each of these techniques is used for training on a sub-sample dataset. We observe that there is a significant decrease in training time when we use local search (almost 7000 times faster), with a negligible decrease in the F_1-score (0.073%).

	Precision	*Recall*	F_1
Hoffmann	75.436	46.615	57.623
MM-Hamming	76.839	50.462	60.918
MM-F-loss	65.991	65.211	65.598

Table 5.3: Overall results on the *positive* dataset.

5.4.3 The Overall Results

Figure 5.4 and Table 5.3 present the overall results of our approaches compared to the baseline on the *positive* dataset. We observe that `MM-F-loss` has an increase in F_1-score to the tune of ∼8% compared to the baseline. This confirms our observation on the sub-sample datasets that we saw earlier.

	Precision	*Recall*	F_β
Hoffmann	75.44	46.62	57.62
MM-F-loss-wt	77.04	53.44	63.11

Table 5.4: Increasing weight on Precision in F_β.

By assigning more weight to precision, we are able to improve over the precision of `Hoffmann` by ∼1.6% (Table 5.4). When precision is tuned with a higher weight during the training of `MM-F-loss`, we see an improvement in precision without much dip in recall.

5.4.4 Discussion

So far we have discussed the performance of various approaches on the *positive* evaluation dataset. Our approach is shown to improve the overall F_β-score that has better recall than the baseline. By suitably tweaking the F_β we show an improvement in precision as well.

The performance of the approaches when evaluated on the entire test dataset (consisting of both nil and non-nil data points) is shown in Table 5.5. Max-margin based approaches generally perform well when trained only on the *positive* dataset, when com-

pared to `Hoffmann`. However, our F_1-scores are $\sim$8% less when we train on the entire dataset that consists of both nil and non-nil data points.

Trained On$\rightarrow$	*entire dataset*	*positive dataset*
Hoffmann	23.14	3.269
MM-Hamming	13.20	16.26
MM-F-loss	13.94	21.93

Table 5.5: F_1-scores on the entire test set.

In a recent work, Xu et al. (2013) provide some statistics about the incompleteness of the Riedel dataset. Out of the sampled 1854 sentences from NYTimes corpus, most of the entity-pairs that express a relation in Freebase correspond to FNs. This is one of the reasons why we do not consider nil labeled data points during training and evaluation.

MIMLRE (Surdeanu et al., 2012) is another state-of-the-art system that is based on the EM algorithm. Since it uses an additional set of features for the relation variables $\mathbf{y}$, it is not our primary baseline. On the *positive* dataset, our model `MM-F-loss` achieves an F_1-score of 65.598% compared to 65.341% of MIMLRE. As part of the future work, we would like to incorporate the additional features present in MIMLRE into our approach.

5.5 Chapter Summary

In this chapter, we have described a novel max-margin approach to optimize non-linear performance measures, such as F_β, in distant supervision of Information Extraction models. Our approach is general and can be applied to other latent variable models in NLP. Our approach involves solving the hard-optimization problem in learning by interleaving Concave-Convex Procedure with dual decomposition. Dual decomposition allowed us to solve the hard sub-problems independently. A key aspect of our approach involves a local-search algorithm, which has led to a speed up of 7,000 times in our experiments. We have demonstrated the efficacy of our approach in distant supervision of relation extraction. Under several conditions, we have shown our technique outperforms very strong baselines, and results in up to 8.5% improvement in F_1-score.

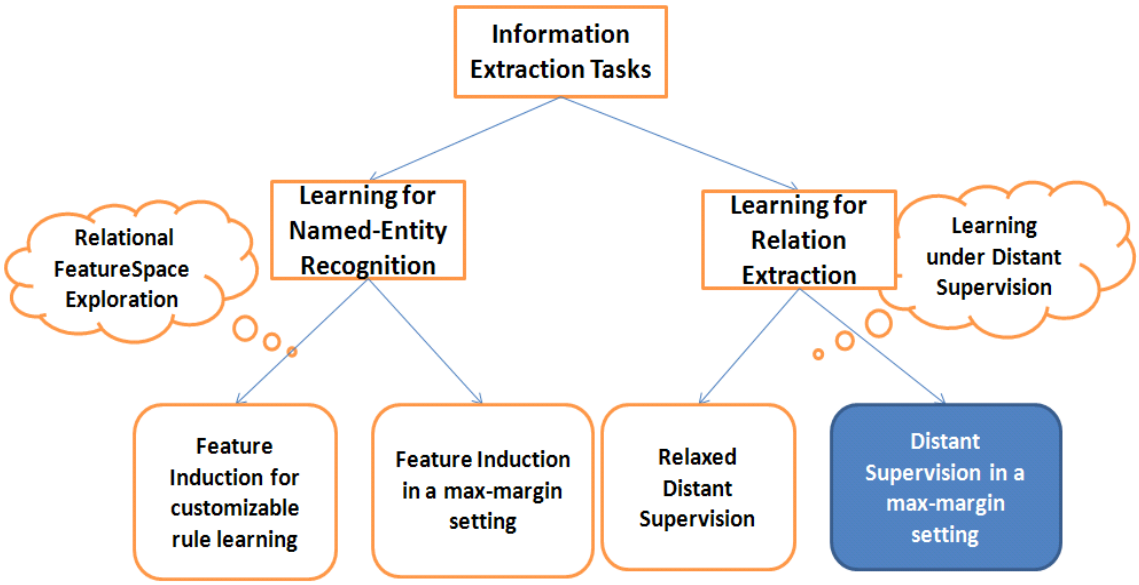


Figure 5.5: Thesis Organization: Chapter 5

Figure 5.5 highlights the part presented in this chapter and the accompanying publication that overlaps with the material presented.

Chapter 6

Conclusion

Our thesis can be summarized as shown in Figure 6.1. The broad theme of each work along with its publication forum is indicated. In the entity extraction setting, we work in the paradigm of *relational feature space exploration*, and in the relation extraction setting, our research has been in the paradigm of *learning under distant supervision*.

In Chapter 3, we presented a system for efficiently inducing named-entity annotation rules. We have designed a feature induction approach that aims to produce accurate rules that can be understood and refined by humans. This has been done by placing special emphasis on low complexity and efficient computation of the induced rules, while mimicking a four-stage approach used for manually constructing rules. The main induction approach consisted of two stages, namely, candidate definition and candidate refinement, while the other two stages, that is, basic feature definition and rule consolidation are assumed to be manually specified. We presented results with both, domain-independent as well as customized basic features. According to our complexity measure, the induced rules have good accuracy and low complexity.

While our complexity measure informs the biases in our system and leads to simpler, smaller extractors, it captures extractor interpretability only to a certain extent. It captures the notion of interpretability only in the feature language. Some of its shortcomings are discussed in Section 3.5.3. Therefore, we believe more work is required to devise a comprehensive quantitative measure for interpretability, and to refine our techniques in order to increase the interpretability of induced rules. We also provide some directions for future research in this area that can have far reaching impact.

In a recent paper, Chiticariu et al. (2013) provide a very interesting analysis of the gulf between industry and academia, in the adoption of rule-based systems. While rule-

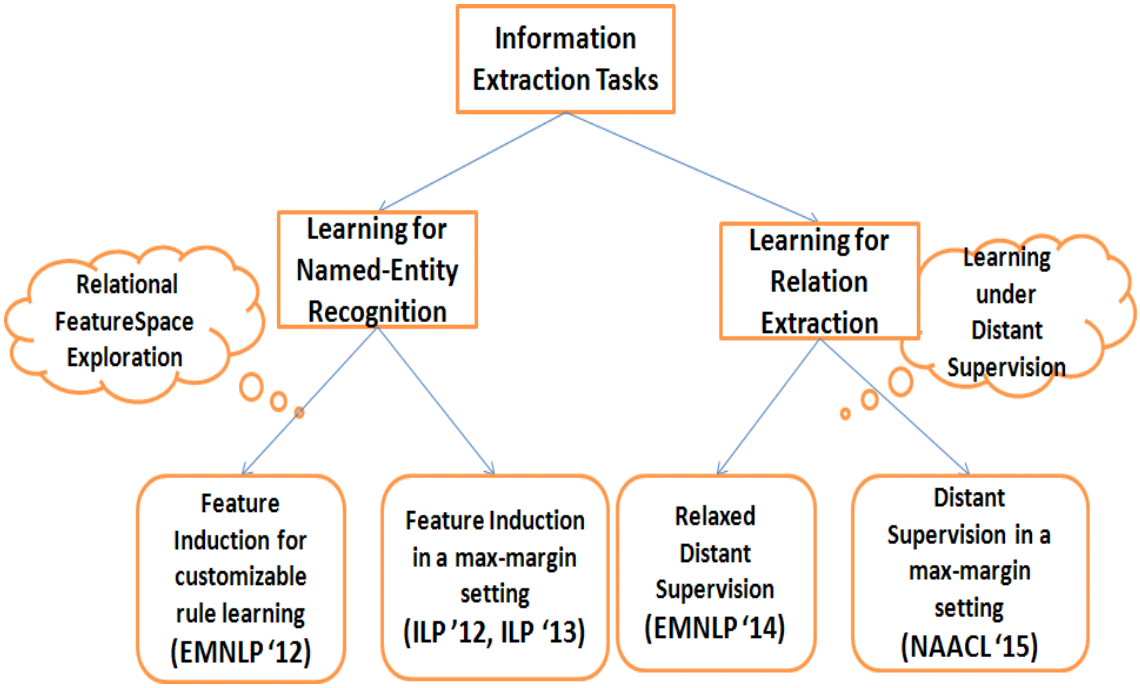


Figure 6.1: Thesis Summary

based systems are extensively used in the industry, the academic community has not been actively engaging to address some of the research challenges in rule-based IE. Our work is a small step to bridge this gap. Our experiment with relational features to train statistical learners reinforces our conviction that rule induction — with emphasis on interpretability — has a lot of potential in building robust IE systems.

Beyond interpretability in the rule language, rules induced should appeal to human intuition. This is out of the scope of the current thesis. It requires careful design of *human-computer interaction* experiments, in order to present the induced rules to a manual rule-developer. This future work can have far reaching implications in addressing the so called *man-machine gap*. Other interesting directions for future work are the introduction of more constructs in our framework, and the application of our techniques to other languages.

In the last couple of years, *distant-supervision based relation extraction* has been a highly active area of research in the IE community. It has the potential to create large training data which is almost impossible to create by human annotators. Although, the data is extremely noisy and needs sophisticated learning algorithms to learn robust classifiers. In Chapter 4, we described an important addition to the model by Hoffmann

et al. (2011) by the use of the *noisy-or* soft constraint, to further relax *the at least one* assumption. Since we posed the inference procedures in Hoffmann using ILP, we could easily add this constraint during the training and inference. Empirically, we showed that the resulting P/R curves have a significant performance boost over Hoffmann's algorithm as a result of this newly added constraint. Although our system has a lower recall when compared to MIMLRE (Surdeanu et al., 2012), it performs competitively with respect to precision at low recall.

Our ILP formulation provides a good framework to add new types of constraints to the problem. In the future, we would like to experiment with other constraints such as modeling the *selectional preferences* of entity types. Additionally, constrains which are global in nature (for example, *a country cannot have more than one head-of-state*) are difficult to enforce, since this would necessitate the interaction across entity pairs. This would also be an interesting direction of future research.

In Chapter 5, we described a novel max-margin approach to optimize non-linear performance measures, such as F_β, in distant supervision of information extraction models. Our approach is general and can be applied to other latent variable models in NLP. Our approach involves solving the hard optimization problem in learning by interleaving the Concave-Convex Procedure with dual decomposition. Dual decomposition allowed us to solve the hard sub-problems independently. A key aspect of our approach involves a local-search algorithm, which has led to a 7,000-time speed-up in our experiments. We have demonstrated the efficacy of our approach in distant supervision of relation extraction. Under several conditions, we have shown that our technique outperforms very strong baselines, and results in an improvement of up to 8.5% in the F_1-score.

Although we solved the hard optimization problem with an efficient dual-decomposition formulation, our algorithms do not scale very well to large datasets. As part of future work, we would like to investigate distributed optimization algorithms as an extension to our solutions. In addition, we would like to maximize other performance measures such as the area under the curve, for information extraction models. We would also like to explore our approach for other latent variable models in NLP, such as those in machine translation.

Appendix A

Relational Feature Class Hierarchy

We state some relationships between the relational feature classes defined in Chapter 2.

Claim A.1. $\mathcal{PF} \subset \mathcal{AF}$.

Proof. From the definition, $\mathcal{PF}$s are $\mathcal{AF}$s with the restriction that a new local variable introduced should be transitively consumed by a single *evidence* predicate. Hence $\mathcal{PF} \subseteq \mathcal{AF}$. Now, consider the clause *a(X) :- b(X,Y), c(Y), d(Y).* Since it follows all the requirements of an $\mathcal{AF}$, it is an absolute feature. However, since there are two *evidence* predicates for the local variable Y, it does not qualify to be a $\mathcal{PF}$. Hence, $\mathcal{PF} \neq \mathcal{AF}$. □

Claim A.2. $\mathcal{AF} \subset \mathcal{CF}$.

Proof. From the definition, $\mathcal{CF}$s are conjunctions of one or more $\mathcal{AF}$s. Therefore, all $\mathcal{AF}$s are $\mathcal{CF}$s (unary conjunctions). Now, consider the $\mathcal{CF}$ clause *a(X) :- b(X,Y), c(Y), b(X,Z), d(Z).* As this a conjunction of two $\mathcal{AF}$s $\big($*a(X) :- b(X,Y), c(Y)* and *a(X) :- b(X,Z), d(Z)*$\big)$, this is not minimal, and hence, not an $\mathcal{AF}$. Hence, $\mathcal{AF} \neq \mathcal{CF}$. □

Claim A.3. $\mathcal{SC} \subset \mathcal{CF}$.

Proof. From the definition, $\mathcal{CF}$s are conjunctions of one or more $\mathcal{AF}$s. Also $\mathcal{SC}$s are conjunctions of evidence predicates at a single sequence position. Since an $\mathcal{SC}$ with a single evidence predicate is also an $\mathcal{AF}$, conjunctions of such single predicate $\mathcal{SC}$s are the same as conjunctions of single predicate $\mathcal{AF}$s, and thus, all $\mathcal{SC}$s are $\mathcal{CF}$s. Now, consider the clauses 3, 5 and 7 which are $\mathcal{CF}$s, but not $\mathcal{SC}$s. Therefore, $\mathcal{SC} \neq \mathcal{CF}$. □

Claim A.4. $\mathcal{CF} \subset \mathcal{DF}$.

Proof. From the definition, $\mathcal{DF}$s are first order definite clauses without any restrictions imposed for $\mathcal{CF}$s. Therefore, $\mathcal{CF} \subseteq \mathcal{DF}$. Now consider the clause *a(X) :- b(X,Y)*, which is a first order relation that does not qualify as a $\mathcal{CF}$, since the variable Y introduced is not reused. Therefore, $\mathcal{CF} \neq \mathcal{DF}$. □

Claim A.5. *Every $\mathcal{AF}$ can be constructed from $\mathcal{PF}$s using unifications.*

Proof. The difference between an $\mathcal{AF}$ and a $\mathcal{PF}$ is that an $\mathcal{AF}$ can have more than one *evidence* predicate for each local variable introduced. Let l_p be a *relational* literal in the body of an $\mathcal{AF}$ clause that introduces only one local variable. Let $l_1, l_2, \ldots, l_{p-1}$ be the set of *relational* literals in the body, which l_p depends on. Let there be $P \geq 0$ number of dependency chains starting from l_p to some *evidence* predicates, each of which is represented as $l^i_{p+1}, \ldots, l^i_k$. We define l_p as a pivot literal if $P > 1$. For simplicity, we assume that there is only one pivot in a clause. Now, we can construct P $\mathcal{PF}$ clauses from this with the body of the i^{th} clause as $l_1, l_2, \ldots, l_{p-1}, l_p, l^i_{p+1}, \ldots, l^i_k$, where l^i_k is an *evidence* predicate. It is trivial to see that these P clauses can be unified to construct the original $\mathcal{AF}$. For multiple pivot literal clauses, the method described above can be applied recursively until $\mathcal{PF}$ clauses are generated. The proof can be extended to pivot literals with multiple new local variables by using a dependency tree structure in place of the chain. □

Claim A.6. *Every $\mathcal{CF}$ can be constructed from $\mathcal{AF}$s by conjunctions.*

Proof. By definition, a clause qualifies as a $\mathcal{CF}$ only if it is constructed from the conjunction of one or more $\mathcal{AF}$s. □

Claim A.7. *$\mathcal{CF}$s are first order $\mathcal{DF}$s with local variable reuse restriction.*

Proof. $\mathcal{AF}$s include maximal clauses generated only with unification (without conjunctions) of $\mathcal{PF}$s. These clauses have the restriction that all local variables introduced need to be consumed transitively. $\mathcal{CF}$s capture all possible conjunctions of $\mathcal{AF}$s, and therefore, can generate any definite clause that is consistent with the local variable consumption restriction. □

Appendix B

SystemT and AQL

SystemT is a declarative Information Extraction system based on an algebraic framework. In SystemT, developers write rules in an *SQL*-like language called *Annotation Query Language (AQL)*. To represent annotations in a document, AQL uses a simple relational data model with three types, namely:

1. *Span*: A region of text within a document identified by its "begin" and "end" positions.
2. *Tuple*: A fixed-size list of spans.
3. *Relation*, or *View*: A multi-set of tuples, where every tuple in the view must be of the same size.

Figure B.1 shows a portion of a *Person* extractor written in AQL. The output of the rules on a sample snippet from a document is shown in Figure B.2.

Basic Constructs in AQL

The basic building block of AQL is a *view*. A *view* is a logical description of a set of tuples in terms of (i) the document text (denoted as a special view called *Document*), and (ii) the contents of other views, as specified in the *from* clauses of each statement. Figure B.1 also illustrates five of the basic constructs that can be used to define a view, which we explain next. We note that this set of constructs form only a subset of the AQL language necessary for the purpose of this work. The complete specification can be found in the AQL manual (IBM, 2012).

```
R1:  create view Caps as
       extract regex /[A-Z](\w|-)+/ on D.text as match from Document D;

R2:  create view First as
       extract dictionary 'FirstNameGazeteer' on D.text as match from Document D;

R3:  create view Last as
       extract dictionary 'LastNameGazeteer' on D.text as match from Document D;

R4:  create view PersonFirst as
       select F.match as match
       from First F, Caps C
       where Equals(F.match, C.match);

R5:  create view PersonFirstLast as
       select CombineSpans(F.match, L.match) as match
       from First F, Last L, Caps C
       where FollowsTok(F.match, L.match, 0, 0) and Equals(L.match, C.match);

R6:  create view PersonCandidate as
       (select R.match from PersonFirst R)
       union all
       (select R.match from PersonFirstLast R);

R7:  create view PersonInvalid as
       select P.match
       from PersonCandidate P, Organization O
       where Overlaps (P.match, O.match);

R8:  create view PersonAll as
       (select R.match from PersonCandidate R)
       minus
       (select R.match from PersonInvalid R);

R9:  create view Person as
       select R.match
       from PersonAll R
       consolidate on R.match using 'ContainedWithin';
```

Complexity of extractor C(E) = 15 + C(Organization)

Figure B.1: Example *Person* extractor in AQL

The extract statement

The *extract* statement specifies basic character-level extraction primitives such as regular expression and dictionary matching over text, creating a tuple for each match. As an

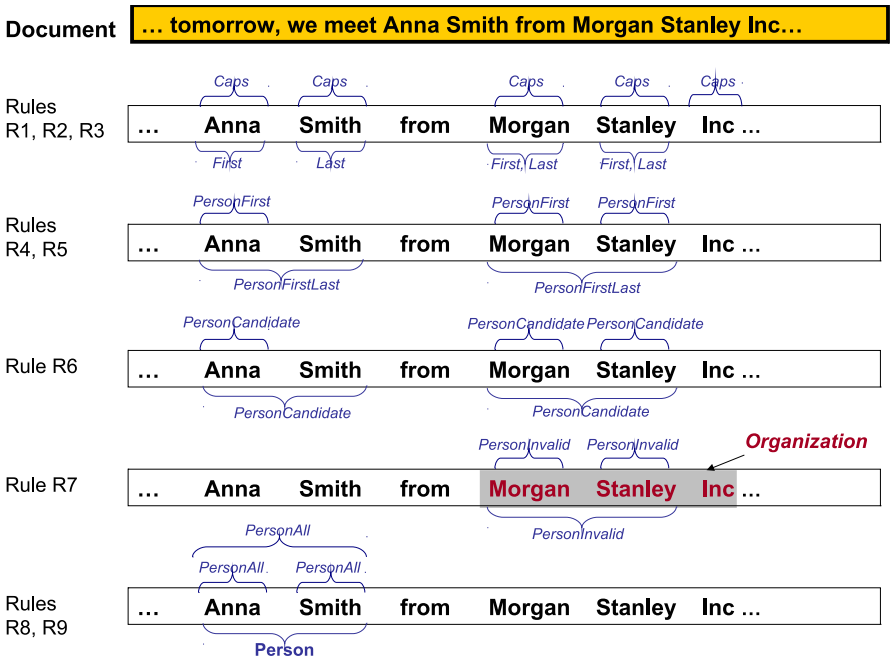


Figure B.2: Output of *Person* extractor on a sample document snippet

example, rule R_1 uses the *extract* statement to identify matches (*Caps* spans) of a regular expression for capitalized words while R_2 and R_3 match gazetteers of common first names (*First* spans) and last names (*Last* spans), respectively.

The select statement

The *select* statement is similar to the SQL select statement, but contains an additional *consolidate on* clause (explained further), along with an extensive collection of text-specific predicates. For example, rule R_4 constructs simple person candidate spans from *First* spans that are also *Caps* spans, in which the equality condition is specified using the *join predicate Equals()*. Rule R_5 illustrates a complex example: it selects *First* spans immediately followed within zero tokens by a *Last* span, where the latter is also a *Caps* span. The two conditions are specified using two join predicates: *FollowsTok* and *Equals*, respectively, which returns true if the span in the first argument is followed by the span in the second argument, within min and max distances measured in tokens (specified by the next two arguments). For each triplet of *First*, *Last* and *Caps* spans that satisfy the two predicates, the *CombineSpans* built-in scalar function in the select clause constructs larger *PersonFirstLast* spans that begin at the begin position of the *First* span, and end

at the end position of the *Last* (also *Caps*) span.

The union all statement

The *union all* statement merges the outputs of two or more statements. For example, rule R_6 unions person candidates identified by rules R_4 and R_5.

The minus statement

The *minus* statement subtracts the output of one statement from the output of another. For example, rule R_8 defines a view *PersonAll* by filtering out *PersonInvalid* tuples from the set of *PersonCandidate* tuples. Notice that rule R_7 used to define the view *PersonInvalid* illustrates another join predicate of AQL called *Overlaps*, which returns true if its two argument spans overlap in the input text. Therefore, at a high level, rule R_8 removes person candidates that overlap with an *Organization* span. (The *Organization* extractor is not depicted in the figure.)

The consolidate clause

The *consolidate* clause of a *select* statement removes selected overlapping spans from the indicated column of the input tuples, according to the specified policy (for instance, "ContainedWithin"). For example, rule R_9 retains *PersonAll* spans that are not contained in other *PersonAll* spans.

Appendix C

Induction Target Language

Our goal is to automatically generate NER extractors with good quality, and at the same time, manageable complexity, so that the extractors can be further refined and customized by the developer. To this end, we focus on inducing extractors using the subset of AQL Constructs described in Section B. We note that we have chosen a small subset of AQL constructs that are sufficient to implement several common operations required for NER. Our experimental results seem to indicate that the subset we consider is sufficient to induce NER extractors with the above properties. However, AQL is a much more expressive language, suitable for general-purpose extraction tasks, and we leave the study of other AQL constructs in the context of rule induction for future work. In this section we describe the building blocks of our target language, and propose a simple definition for measuring the complexity of an extractor.

The components of the target language are as follows, summarized in Table C.1.

Basic features (BF)

BF features are specified using the *extract* statement, such as rules R_1 to R_3 in Figure B.1. In this work, we assume as input a set of *basic features* consist of dictionaries and regular expressions.

Candidate definition (CD)

CD features are expressed using the *select* statement to combine BF features with join predicates (for example, *Equals*, *FollowsTok* or *Overlaps*), and the *CombineSpans* scalar function to construct larger candidate spans from input spans. Rules R_4 and R_5 in Figure B.1 are example CD rules. In general, a CD view is defined as: ‘‘`Select`

Phase name	AQL statements	Prescription	Rule Type
Basic Features	`extract`	Off-the-shelf, Learning using prior work Riloff (1993); Li et al. (2008)	Basic Features Definition
Phase 1 (Clustering and RLGG)	`select`	Bottom-up learning (LGG), Top-down refinement	Development of Candidate Rules
Phase 2 (Propositional Rule Learning)	`select, union all, minus`	RIPPER, Lightweight Rule Induction	Candidate Rules Filtering
Consolidation	`consolidate, union all`	Manually identified consolidation rules, based on domain knowledge	Consolidation rules

Table C.1: Phases in induction, the language constructs invoked in each phase, the prescriptions for inducing rules in the phase, and the corresponding type of rule in manual rule-development.

`all spans constructed from view`$_1$`, view`$_2$`, ..., view`$_n$`, such that all join predicates are satisfied.''`

Candidate refinement (CR)

CR features are used to discard spans output by the CD features that may be incorrect. In general, a CR feature is defined as: ``` ``From the list of spans of view`$_{valid}$ `subtract all those spans that belong to view`$_{invalid}$`'' ```. `view`$_{valid}$ is obtained by joining all the positive CD clues on the *Equals* predicate and `view`$_{invalid}$ is obtained by joining all the negative overlapping clues with the *Overlaps* predicate and subsequently 'union'ing all the negative clues. (for example, similar in spirit to rules R_6, R_7 and R_8 in Figure B.1, except that the subtraction is done from a single view and not the *union* of multiple views).

Consolidation (CO)

Finally, a *select* statement with a fixed *consolidate* clause is used for each entity type to remove overlapping spans from CR rules. An example CO view is defined by rule R_9 in Figure B.1.

Bibliography

Abiteboul, S., Hull, R., Vianu, V. (Eds.), 1995. Foundations of Databases: The Logical Level, 1st Edition. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Appelt, D. E., Onyshkevych, B., 1998. The common pattern specification language. In: Proceedings of a Workshop on Held at Baltimore, Maryland: October 13-15, 1998. TIPSTER '98. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 23–30.
URL http://dx.doi.org/10.3115/1119089.1119095

Bach, F. R., 2009. High-dimensional non-linear variable selection through hierarchical kernel learning. CoRR abs/0909.0844.
URL http://arxiv.org/abs/0909.0844

Bender, O., Och, F. J., Ney, H., 2003. Maximum entropy models for named entity recognition. In: Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4. CONLL '03. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 148–151.
URL http://dx.doi.org/10.3115/1119176.1119196

Bunescu, R., Pasca, M., April 2006. Using encyclopedic knowledge for named entity disambiguation. In: Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06), Trento, Italy. pp. 9–16.
URL http://www.cs.utexas.edu/~ml/publication/paper.cgi?paper=encyc-eacl-06.ps.gz

Califf, M. E., August 1998. Relational learning techniques for natural language information extraction. Ph.D. thesis, Department of Computer Sciences, University of Texas, Austin, TX, also appears as Artificial Intelligence Laboratory Technical Report AI

98-276.
URL http://www.cs.utexas.edu/users/ai-lab/?califf:thesis98

Califf, M. E., Mooney, R. J., August 1997. Applying ilp-based techniques to natural language information extraction: An experiment in relational learning. In: Workshop Notes of the IJCAI-97 Workshop on Frontiers of Inductive Logic Programming. Nagoya, Japan, pp. 7–11.
URL http://www.cs.utexas.edu/users/ai-lab/?califf:ijcai:filp97

Califf, M. E., Mooney, R. J., 1999. Relational learning of pattern-match rules for information extraction. In: Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence. AAAI '99/IAAI '99. American Association for Artificial Intelligence, Menlo Park, CA, USA, pp. 328–334.
URL http://dl.acm.org/citation.cfm?id=315149.315318

Chiticariu, L., Chu, V., Dasgupta, S., Goetz, T. W., Ho, H., Krishnamurthy, R., Lang, A., Li, Y., Liu, B., Raghavan, S., Reiss, F. R., Vaithyanathan, S., Zhu, H., 2011. The systemt ide: an integrated development environment for information extraction rules. In: Proceedings of the 2011 international conference on Management of data. SIGMOD '11. ACM, New York, NY, USA, pp. 1291–1294.
URL http://doi.acm.org/10.1145/1989323.1989479

Chiticariu, L., Krishnamurthy, R., Li, Y., Raghavan, S., Reiss, F. R., Vaithyanathan, S., 2010a. Systemt: An algebraic approach to declarative information extraction. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. ACL '10. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 128–137.
URL http://dl.acm.org/citation.cfm?id=1858681.1858695

Chiticariu, L., Krishnamurthy, R., Li, Y., Reiss, F., Vaithyanathan, S., 2010b. Domain adaptation of rule-based annotators for named-entity recognition tasks. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. EMNLP '10. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1002–1012.

URL http://dl.acm.org/citation.cfm?id=1870658.1870756

Chiticariu, L., Li, Y., Reiss, F. R., 2013. Rule-based information extraction is dead! long live rule-based information extraction systems! In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL. pp. 827–832.
URL http://aclweb.org/anthology/D/D13/D13-1079.pdf

Ciravegna, F., 2001. $(LP)^2$, an adaptive algorithm for information extraction from web-related texts. In: Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining.
URL http://citeseer.ist.psu.edu/481342.html

Cohen, W. W., 1995. Fast effective rule induction. In: Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995. pp. 115–123.
URL http://dblp.uni-trier.de/rec/bib/conf/icml/Cohen95

Craven, M., Kumlien, J., 1999. Constructing biological knowledge bases by extracting information from text sources. In: Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology. AAAI Press, pp. 77–86.
URL http://dl.acm.org/citation.cfm?id=645634.663209

Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Aswani, N., Roberts, I., Gorrell, G., Funk, A., Roberts, A., Damljanovic, D., Heitz, T., Greenwood, M. A., Saggion, H., Petrak, J., Li, Y., Peters, W., 2011. Text Processing with GATE (Version 6).
URL http://tinyurl.com/gatebook

Felzenszwalb, P. F., Girshick, R. B., McAllester, D. A., Ramanan, D., 2010. Object detection with discriminatively trained part-based models. IEEE Trans. Pattern Anal. Mach. Intell. 32 (9), 1627–1645.
URL http://doi.ieeecomputersociety.org/10.1109/TPAMI.2009.167

Finkel, J. R., Manning, C. D., 2009. Nested named entity recognition. In: Proceedings of

the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1. EMNLP '09. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 141–150.
URL http://dl.acm.org/citation.cfm?id=1699510.1699529

Flach, P., Lachiche, N., 1999. 1bc: A first-order bayesian classifier. In: Deroski, S., Flach, P. (Eds.), Inductive Logic Programming. Vol. 1634 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 92–103.
URL http://dx.doi.org/10.1007/3-540-48751-4_10

Florian, R., Ittycheriah, A., Jing, H., Zhang, T., 2003. Named entity recognition through classifier combination. In: Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4. CONLL '03. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 168–171.
URL http://dx.doi.org/10.3115/1119176.1119201

Forney, G. D., March 1973. The viterbi algorithm. Proceedings of the IEEE 61, 268 – 278.
URL http://www.bibsonomy.org/bibtex/27b2ae314c64d34ca0cf52425ea541f7d/bfields

Fürnkranz, J., Feb. 1999. Separate-and-conquer rule learning. Artif. Intell. Rev. 13 (1), 3–54.
URL http://dx.doi.org/10.1023/A:1006524209794

Fürnkranz, J., Widmer, G., 1994. Incremental reduced error pruning. In: Hirsh, W. W. C. (Ed.), Machine Learning Proceedings 1994. Morgan Kaufmann, San Francisco (CA), pp. 70 – 77.
URL http://www.ai.univie.ac.at/~juffi/publications/ml-94.ps.gz

Gaines, B. R., Compton, P., Nov. 1995. Induction of ripple-down rules applied to modeling large databases. J. Intell. Inf. Syst. 5 (3), 211–228.
URL http://dx.doi.org/10.1007/BF00962234

GuoDong, Z., Jian, S., Jie, Z., Min, Z., 2005. Exploring various knowledge in relation extraction. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. ACL '05. Association for Computational Linguistics, Stroudsburg,

PA, USA, pp. 427–434.
URL http://dx.doi.org/10.3115/1219840.1219893

Hoffmann, R., Zhang, C., Ling, X., Zettlemoyer, L., Weld, D. S., 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1. HLT '11. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 541–550.
URL http://dl.acm.org/citation.cfm?id=2002472.2002541

Horn, A., 1951. On sentences which are true of direct unions of algebras. The Journal of Symbolic Logic 16 (1), pp. 14–21.
URL http://www.jstor.org/stable/2268661

IBM, 2012. IBM InfoSphere BigInsights - Annotation Query Language (AQL) reference.
URL http://www-01.ibm.com/support/knowledgecenter/SSPT3X_2.0.0/com.ibm.swg.im.infosphere.biginsights.text.doc/doc/biginsights_aqlref_con_aql-overview.html

Jansche, M., Abney, S. P., 2002. Information extraction from voicemail transcripts. In: Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10. EMNLP '02. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 320–327.
URL http://dx.doi.org/10.3115/1118693.1118734

Jawanpuria, P., Nath, J. S., Ramakrishnan, G., 2011. Efficient rule ensemble learning using hierarchical kernels. In: Getoor, L., Scheffer, T. (Eds.), ICML. Omnipress, pp. 161–168.
URL http://dblp.uni-trier.de/db/conf/icml/icml2011.html

Joachims, T., 2005. A support vector method for multivariate performance measures. In: Proceedings of the 22nd International Conference on Machine Learning. ICML '05. ACM, New York, NY, USA, pp. 377–384.
URL http://doi.acm.org/10.1145/1102351.1102399

Keshet, J., 2014. Optimizing the measure of performance in structured prediction. In:

Nowozin, S., Gehler, P. V., Jancsary, J., Lampert, C. H. (Eds.), Advanced Structured Prediction. The MIT Press.
URL http://u.cs.biu.ac.il/~jkeshet/papers/Keshet14.pdf

Komodakis, N., Paragios, N., Tziritas, G., Mar. 2011. Mrf energy minimization and beyond via dual decomposition. IEEE Trans. Pattern Anal. Mach. Intell. 33 (3), 531–552.
URL http://dx.doi.org/10.1109/TPAMI.2010.108

Kripke, S. A., 1980. Naming and Necessity. Harvard University Press.

Krupka, G. R., Hausman, K., 1998. Isoquest: Description of the netowl^{TM} extractor system as used in muc-7. In: Proceedings of the Seventh Message Understanding Conference (MUC-7).
URL http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/muc_7_proceedings/isoquest.pdf

Kulkarni, S., Singh, A., Ramakrishnan, G., Chakrabarti, S., 2009. Collective annotation of wikipedia entities in web text. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '09. ACM, New York, NY, USA, pp. 457–466.
URL http://doi.acm.org/10.1145/1557019.1557073

Lafferty, J. D., McCallum, A., Pereira, F. C. N., 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning. ICML '01. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 282–289.
URL http://dl.acm.org/citation.cfm?id=645530.655813

Li, Y., Krishnamurthy, R., Raghavan, S., Vaithyanathan, S., Jagadish, H. V., 2008. Regular expression learning for information extraction. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. EMNLP '08. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 21–30.
URL http://dl.acm.org/citation.cfm?id=1613715.1613719

Liu, B., Chiticariu, L., Chu, V., Jagadish, H. V., Reiss, F. R., Sep. 2010. Automatic rule refinement for information extraction. Proc. VLDB Endow. 3 (1-2), 588–597.

URL http://dx.doi.org/10.14778/1920841.1920916

Maynard, D., Bontcheva, K., Cunningham, H., 2003. Towards a semantic extraction of named entities. In: Recent Advances in Natural Language Processing.
URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.120.8162

McCallum, A., Li, W., 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In: Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4. CONLL '03. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 188–191.
URL http://dx.doi.org/10.3115/1119176.1119206

McCreath, E., Sharma, A., 1998. Lime: A system for learning relations. In: Proceedings of the 9th International Conference on Algorithmic Learning Theory. ALT '98. Springer-Verlag, London, UK, pp. 336–374.
URL http://dl.acm.org/citation.cfm?id=647716.735752

Mintz, M., Bills, S., Snow, R., Jurafsky, D., 2009. Distant supervision for relation extraction without labeled data. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL. ACL '09. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1003–1011.
URL http://dl.acm.org/citation.cfm?id=1690219.1690287

Muggleton, S., Feng, C., 1990. Efficient induction of logic programs. In: Proceedings of the First Conference on Algorithmic Learning Theory. Tokyo, Japan, pp. 368–381.
URL http://dblp.uni-trier.de/rec/bib/conf/alt/MuggletonF90

Nadeau, D., Sekine, S., Jan. 2007. A survey of named entity recognition and classification. Lingvisticae Investigationes 30 (1), 3–26.
URL http://www.ingentaconnect.com/content/jbp/li/2007/00000030/00000001/art00002

Nagesh, A., Nair, N., Ramakrishnan, G., 2013. Comparison between explicit learning and implicit modeling of relational features in structured output spaces. In: Late Breaking Papers of the 23rd International Conference on Inductive Logic Programming, Rio de

Janeiro, Brazil, August 28th - to - 30th, 2013. pp. 29–39.
URL http://ceur-ws.org/Vol-1187/paper-06.pdf

Nair, N., Nagesh, A., Ramakrishnan, G., 2012a. Probing the space of optimal markov logic networks for sequence labeling. In: Inductive Logic Programming - 22nd International Conference, ILP 2012, Dubrovnik, Croatia, September 17-19, 2012, Revised Selected Papers. pp. 193–208.
URL http://dx.doi.org/10.1007/978-3-642-38812-5_14

Nair, N., Ramakrishnan, G., Krishnaswamy, S., 2011. Enhancing activity recognition in smart homes using feature induction. In: Proceedings of the 13th international conference on Data warehousing and knowledge discovery. DaWaK'11. Springer-Verlag, Berlin, Heidelberg, pp. 406–418.
URL http://dl.acm.org/citation.cfm?id=2033616.2033657

Nair, N., Saha, A., Ramakrishnan, G., Krishnaswamy, S., 2012b. Rule ensemble learning using hierarchical kernels in structured output spaces. In: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.
URL http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/4918

Ng, A. Y., Jordan, M. I., 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In: Dietterich, T., Becker, S., Ghahramani, Z. (Eds.), Advances in Neural Information Processing Systems 14. MIT Press, pp. 841–848.
URL http://papers.nips.cc/paper/2020-on-discriminative-vs-generative-classifier
pdf

Nienhuys-Cheng, S.-H., Wolf, R. d., 1997. Foundations of Inductive Logic Programming. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Patel, A., Ramakrishnan, G., Bhattacharya, P., 2010. Incorporating linguistic expertise using ilp for named entity recognition in data hungry indian languages. In: De Raedt, L. (Ed.), Inductive Logic Programming. Vol. 5989 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 178–185.

URL http://dx.doi.org/10.1007/978-3-642-13840-9_16

Plotkin, G., 1970. A note on inductive generalization. In: Machine Intelligence. Vol. 5. Edinburgh University Press, pp. 153–163.

Rabiner, L. R., 1990. Readings in speech recognition. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, Ch. A tutorial on hidden Markov models and selected applications in speech recognition, pp. 267–296.
URL http://dl.acm.org/citation.cfm?id=108235.108253

Ranjbar, M., Lan, T., Wang, Y., Robinovitch, S. N., Li, Z.-N., Mori, G., April 2013. Optimizing nondecomposable loss functions in structured prediction. IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (4), 911–924.
URL http://doi.ieeecomputersociety.org/10.1109/TPAMI.2012.168

Ranjbar, M., Vahdat, A., Mori, G., 2012. Complex loss optimization via dual decomposition. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012. pp. 2304–2311.
URL http://dx.doi.org/10.1109/CVPR.2012.6247941

Reiss, F., Raghavan, S., Krishnamurthy, R., Zhu, H., Vaithyanathan, S., 2008. An algebraic approach to rule-based information extraction. In: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering. ICDE '08. IEEE Computer Society, Washington, DC, USA, pp. 933–942.
URL http://dx.doi.org/10.1109/ICDE.2008.4497502

Riedel, S., Yao, L., McCallum, A., 2010. Modeling relations and their mentions without labeled text. In: Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part III. ECML PKDD'10. Springer-Verlag, Berlin, Heidelberg, pp. 148–163.
URL http://dl.acm.org/citation.cfm?id=1889788.1889799

Riloff, E., 1993. Automatically constructing a dictionary for information extraction tasks. In: Proceedings of the Eleventh National Conference on Artificial Intelligence. AAAI'93. AAAI Press, pp. 811–816.
URL http://dl.acm.org/citation.cfm?id=1867270.1867391

Ritter, A., Zettlemoyer, L., Mausam, Etzioni, O., 2013. Modeling missing data in distant supervision for information extraction. TACL 1, 367–378.
URL http://www.transacl.org/wp-content/uploads/2013/10/paperno30.pdf

Rosenfeld, N., Meshi, O., Globerson, A., Tarlow, D., 2014. Learning structured models with the auc loss and its generalizations. In: Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS).
URL http://research.microsoft.com/apps/pubs/default.aspx?id=208393

Roth, D., Yih, W., 2001. Relational learning via propositional algorithms: An information extraction case study. In: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001. pp. 1257–1263.

Roth, D., Yih, W., 2004. A linear programming formulation for global inference in natural language tasks. In: Proceedings of the Eighth Conference on Computational Natural Language Learning, CoNLL 2004, Held in cooperation with HLT-NAACL 2004, Boston, Massachusetts, USA, May 6-7, 2004. pp. 1–8.
URL http://aclweb.org/anthology/W/W04/W04-2401.pdf

Rush, A. M., Collins, M., 2012. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. J. Artif. Intell. Res. (JAIR) 45, 305–362.
URL http://dx.doi.org/10.1613/jair.3680

Sandhaus, E., 2008. The New York Times Annotated Corpus. Linguistic Data Consortium, Philadelphia 6 (12).
URL https://catalog.ldc.upenn.edu/LDC2008T19

Sarawagi, S., Mar. 2008. Information extraction. Found. Trends databases 1 (3), 261–377.
URL http://dx.doi.org/10.1561/1900000003

Sekine, S., Nobata, C., 2004. Definition, dictionaries and tagger for extended named entity hierarchy. In: Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004, May 26-28, 2004, Lisbon, Portugal.
URL http://www.lrec-conf.org/proceedings/lrec2004/pdf/65.pdf

Singh, S., Hillard, D., Leggetter, C., 2010. Minimally-supervised extraction of entities from text advertisements. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. HLT '10. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 73–81.
URL http://dl.acm.org/citation.cfm?id=1857999.1858008

Soderland, S., Feb. 1999. Learning information extraction rules for semi-structured and free text. Mach. Learn. 34 (1-3), 233–272.
URL http://dx.doi.org/10.1023/A:1007562322031

Srihari, R., Niu, C., Li, W., 2000. A hybrid approach for named entity and sub-type tagging. In: Proceedings of the Sixth Conference on Applied Natural Language Processing. ANLC '00. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 247–254.
URL http://dx.doi.org/10.3115/974147.974181

Srinivas, S., 1993. A generalization of the noisy-or model. In: Proceedings of the Ninth International Conference on Uncertainty in Artificial Intelligence. UAI'93. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 208–215.
URL http://dl.acm.org/citation.cfm?id=2074473.2074499

Surdeanu, M., Ciaramita, M., Mar. 2007. Robust information extraction with perceptrons. In: Proceedings of the NIST 2007 Automatic Content Extraction Workshop (ACE07).
URL http://www.surdeanu.name/mihai/papers/ace07a.pdf

Surdeanu, M., Gupta, S., Bauer, J., McClosky, D., Chang, A. X., Spitkovsky, V. I., Manning, C. D., November 2011. Stanford's distantly-supervised slot-filling system. In: Proceedings of the Fourth Text Analysis Conference (TAC 2011). Gaithersburg, Maryland, USA.
URL pubs/kbp2011-slotfilling.pdf

Surdeanu, M., Tibshirani, J., Nallapati, R., Manning, C. D., 2012. Multi-instance multi-label learning for relation extraction. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. EMNLP-CoNLL '12. Association for Computational Linguistics,

Stroudsburg, PA, USA, pp. 455–465.
URL http://dl.acm.org/citation.cfm?id=2390948.2391003

Tarlow, D., Zemel, R. S., 2012. Structured output learning with high order loss functions. In: Lawrence, N. D., Girolami, M. A. (Eds.), Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS-12). Vol. 22. pp. 1212–1220.
URL http://jmlr.csail.mit.edu/proceedings/papers/v22/tarlow12a/tarlow12a.pdf

Taskar, B., Guestrin, C., Koller, D., 2003. Max-margin markov networks. In: Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]. pp. 25–32.
URL http://papers.nips.cc/paper/2397-max-margin-markov-networks

Tjong Kim Sang, E. F., De Meulder, F., 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In: Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4. CONLL '03. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 142–147.
URL http://dx.doi.org/10.3115/1119176.1119195

Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y., 2004. Support vector machine learning for interdependent and structured output spaces. In: Proceedings of the Twenty-first International Conference on Machine Learning. ICML '04. ACM, New York, NY, USA, p. 104.
URL http://doi.acm.org/10.1145/1015330.1015341

Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y., Dec. 2005. Large margin methods for structured and interdependent output variables. J. Mach. Learn. Res. 6, 1453–1484.
URL http://dl.acm.org/citation.cfm?id=1046920.1088722

van Rijsbergen, C. J., 1979. Information retrieval, 2nd Edition. Butterworths, London.
URL http://www.dcs.gla.ac.uk/Keith/Preface.html

Wang, Y., Mori, G., 2011. Hidden part models for human action recognition: Probabilistic versus max margin. IEEE Trans. Pattern Anal. Mach. Intell. 33 (7), 1310–1323.
URL http://doi.ieeecomputersociety.org/10.1109/TPAMI.2010.214

Witten, I. H., Frank, E., Hall, M. A., 2011. Data Mining: Practical Machine Learning Tools and Techniques, 3rd Edition. Morgan Kaufmann, Amsterdam.

Xu, W., Hoffmann, R., Zhao, L., Grishman, R., August 2013. Filling knowledge base gaps for distant supervision of relation extraction. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Association for Computational Linguistics, Sofia, Bulgaria, pp. 665–670.
URL http://www.aclweb.org/anthology/P13-2117

Yao, L., Riedel, S., McCallum, A., 2010. Collective cross-document relation extraction without labelled data. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. EMNLP '10. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1013–1023.
URL http://dl.acm.org/citation.cfm?id=1870658.1870757

Yu, C. J., Joachims, T., 2009. Learning structural svms with latent variables. In: Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009. p. 147.
URL http://doi.acm.org/10.1145/1553374.1553523

Yuille, A. L., Rangarajan, A., Apr. 2003. The concave-convex procedure. Neural Comput. 15 (4), 915–936.
URL http://dx.doi.org/10.1162/08997660360581958