CONSTRAINT-BASED METRIC-AWARE APPROACH FOR RELATION CO-EXTRACTION

BY

XIAOYU CHEN

THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science in the Graduate College of the University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Adviser:

Professor Kevin Chen-Chuan Chang
This thesis focuses on relation extraction within unstructured text data. We are interested in the bootstrapping approach, in which only a small portion of examples are given to train the extractor. The training of the extractor is actually a process of finding good textual representation patterns for that relationship and the duality relationship between tuples and patterns are explored as a mutual enhancement in an iterative way. However, due to the lack of decent amount of labelled data at the beginning, the bootstrapping performance is often unsatisfactory. Recent literatures explore additional meta level information such as constraints and find a way to add it along with bootstrapping seeds to further reinforce supervision. Our approach takes a step further by exploring how to better incorporate such domain specific constraints into the ranking process of selecting textual patterns for better extraction precision and recall. Thus, we call it a constraint-based metric-aware approach. We explore three types of general constraints and develop models for each of them. We finally conduct experiment on the Wikipedia article dataset, and the results show that with our model, we can achieve significant performance boost in terms of f1 score.
To all the people around me, for their help and support.
Acknowledgments

I have been working on this thesis for more than a year, which is the crystallization of both my sweat and the generous help from all the people around me.

In particular, I would like to thank my advisor, Kevin Chen-Chuan Chang, for his support, patience, and trust. It was him who first brought up the whole setting and basic idea in this thesis, which gave me the very sparkling light I needed to start with everything else. We discussed and met through these two years on a regular weekly basis and what I got from him is the kindly guidance towards the area of entity-centric information extraction. Due to his encouragement and advice, I gained great knowledge by reading papers, taking related courses and conducting experiments. I can see how I started with deficient background and continued growing steadily to become a competent researcher.

For sure, the road towards success can never be easy. There was one time when I wandered into a dead end and struggled for the light of hope. I am really thankful for my parents, my girlfriend, who told me to start over and try again. I am glad that I was able to endure the adversity and get back on my feet to fight with perseverance.

Last but not least are my best friends and lab mates here, Jiahui Jiang, Mianwei Zhou, Rui Li, Yuan Fang, Mangesh Bendre, Long Pham, Jia Wang, Vincent Zheng, Ji Yang, Zequn Zhang. They are of great influence to me with my life and study here. These all together form the best memories for me in UIUC that are worth to be remembered for a lifetime.
# Table of Contents

Chapter 1  Introduction and Background .......................... 1  
  1.1 Introduction ......................................... 1  
  1.2 Related Work ....................................... 3  

Chapter 2  Problem and Model Framework Description ............ 6  
  2.1 Problem Settings .................................... 6  
  2.2 General Constraint Types ............................. 7  
  2.3 Iterative Bootstrapping .............................. 7  
  2.4 Ranking Model: TCP Graph ......................... 8  

Chapter 3  Inference with Mutual Exclusion Constraint .......... 11  

Chapter 4  Inference With Cardinality Constraint ............... 13  
  4.1 Basic Guiding Principles ........................... 13  
  4.2 Extending Label Set ................................ 15  
  4.3 Precision Inference Derivation ..................... 19  

Chapter 5  Inference With Evidence-Path Agreement Constraint . 21  
  5.1 Evidence Path ........................................ 21  
  5.2 Different Ways To Extract .......................... 22  
  5.3 Relation Expressed In Path-wise Way ............... 23  
  5.4 Derivation Detail .................................... 27  
  5.5 Interpretation of Evidence Path Inference ........... 28  

Chapter 6  Experiments and Empirical Study ..................... 30  
  6.1 Basic Experiment Settings ........................... 30  
  6.2 Overall Performance ................................ 32  
  6.3 Discussion .......................................... 34  
  6.4 Conclusion and Future Work ......................... 35  

References .............................................. 36
1.1 Introduction

Relation extraction has been studied for a long time. This is a key technique for building relational databases from unstructured or semi-structured datasets like the web. Since most of nowadays’ relationships are stated in natural language as text, so extracting a specific relation needs finding good natural language pattern representations of that relation.

A well-known approach is discussed in Snowball [1], where the duality of patterns and tuples is discovered and can be automatically extracted using only a small number of tuples. It is basically a bootstrapping learning algorithm where the initial inputs are just a few tuples of the target relationship and it will iteratively extract new patterns and new tuples. PRDualRank [2] is another similar approach that provides better ranking and selection for patterns. This paper will be focusing on this kind of approach.

This kind of weakly-supervised bootstrapping approach will often suffer from low accuracy, known as Semantic Drift [3]. To help improve this issue while keeping weak supervision, Andrew et al. [4] propose a way to couple several target relation extractor together and through constraints for filtering between them, it will boost up each other’s precision.

While this kind of constraint-based coupled bootstrapping has been studied before [4] [5] [3], they are not metric-aware. That is, they separate the constraint-based filtering process from metrics evaluation process and the evaluation metrics are often hidden from the constraint filtering process (e.g. filter
before evaluation). In this paper, we propose a systematic way to combine them in a single process, that is both metric-aware and constraints-aware for bootstrapping relation co-extraction.

Our key observation is that, constraints should be specified within evaluation metrics (precision and recall) in a probabilistic way. In weakly supervised bootstrapping systems, precision and recall for tuples and patterns denote the confidence for extracting them. Thus, tuples should be treated differently according to their metrics in constraint-based filtering (e.g. violating the constraints against a single high confidence tuple may even have higher chances of being filtered out than violating against a set of low confidence tuples). On the other hand, evaluation of precision and recall should take into account the constraints as well. Constraints are like high-level supervision given by the user according to domain knowledge, and should be treated in a similar fashion as the instance level supervision (i.e. labels) given as input. To be specific, we want to incorporate constraints into the inference process for precision and recall.

Figure 1.2 gives a overview of our whole framework. In summary, this paper makes the following contributions:

- We propose a systematic way to combine constraints within the inference process of evaluation metrics (precision and recall) in relation extraction.
Figure 1.2: Illustration of our framework overview, which mainly includes two parts: iterative bootstrapping extraction process and constraint-based ranking. *TCP graph* is a tripartite graph consists of tuple, context and patterns, which will be the graph for pattern evaluation with precision and recall as metrics. Our framework can couple different relationships together and provide a constraint-based metric-aware approach for ranking the tuples and patterns on the graph.

- We implement a bootstrapping system to couple the different relation types together and solve three general constraint types using a single unified random walk based inference framework.

- We discover and provide a way to evaluate path-wise extraction through evidence path, a novel tuple extraction way by joining tuples along a path of several different relation types.

1.2 Related Work

Extracting the tuples of a relation type given a large text corpus has long been studied. With large labeled data, we can apply machine learning techniques and define meaningful features to find patterns or directly label the data. This is discussed in [6, 7] as Distant Supervision. Some recent distant supervision work tend to couple relations together to jointly infer the labeling as in [8, 9], and show some improvement in extraction results. But constructing such labeled data is manual intensive and since our goal is to extract this relation,
we cannot assume we have such results beforehand. Besides, since different relationship may differ in its concept complexity (VC dimension), it’s hard to find the perfect size of training set and apply the model for every relationship. On the other hand, bootstrapping learning methods \[1, 2\] as discussed before, can greatly reduce the amount of manual labels but are all vulnerable to semantic drift.

To help boost up the precision with weak supervision, several collaborative extraction approaches have been put forward in literature \[4, 5, 3\]. Work \[3\] developed on \[5\], and they tried to co-extract several types of unary relations together. They made strong assumption that a noun phrase can only belong to one category, and use this as a constraint to enhance the precision of the extraction. For example, an extracted entity cannot be both a person and an organization. Though this strong assumption may not be true at all times, their results show the insight of this technique. In \[4\], it coupled the category extractor (unary relation extractor) with relation extractor to enhance precision. They gave three rules as general constraints between different extractors and leverage these constraints to enhance precision. For example, when doing binary relation extraction, they would check the types of the entities extracted from unary relations. These ideas are similar to our approach but they didn’t explore the semantic meaning and connections between each relation (e.g. StarIn(Leonardo, Titanic) with StarIn(Kate, Titanic) will imply Costar(Leonardo, Kate)). They were simply coupling relations together. Furthermore, they separate the process of constraint-based filtering and metrics evaluation. But in our work, we explore the multiple ways of expressing a relation through meta-paths on the schema graph and provide a systematic way to specify and evaluate the constraints using precision and recall.

Evaluating patterns for better extraction is the key for getting higher f1 score iteratively. Our evaluation framework is based on \[2\], which is a semi-supervised learning as label propagation framework \[10, 11, 10, 20, 21, 22\], in which all the instances are put onto a graph, and the edge are weighted based
on some kind of measurement. The goal is to infer the unlabeled nodes, using the existing labels and structure of the graph. We further develop our model by enabling propagation across different relations by cross-relation connections between tuples.

There are some literatures on the concept and ways to utilize meta-path \cite{12, 13}. In terms of the concept of meta-path, we use the same definition as in \cite{12}. However, in terms of utilizing the meta-path, we are different from \cite{13} because they are using as features topological structures (e.g. how many paths are there connecting two entities), but we are using patterns (i.e. the natural language that states the relationship). There are also many other differences, such as the task definition, learning and inference framework etc.

Recent research in information extraction explores a new type of relation extraction called Open Information Extraction \cite{14, 15}. Traditional approaches to IE does not scale to corpora where the number of target relations is very large, or where the target relations cannot be specified in advance. Open IE solves this problem by identifying relation phrases – phrases that denote relations in English sentences. The automatic identification of relation phrases enables the extraction of arbitrary relations from sentences, obviating the restriction to a pre-specified vocabulary. There have been lots of systems \cite{14, 16, 17, 18, 19} in this area and the performance is getting better and better.
Chapter 2

Problem and Model Framework Description

We are focused on solving constraints between tuples (instance level) or between relations (concept level). Before going deep into the types of constraints, we give formal definitions of our settings.

2.1 Problem Settings

A schema graph $G^S = (EN, R)$ is given to denote the target relations we are interested in extracting, where $EN$ is a set of entity types and $R$ is the edge set of the graph used to denote the relationship between entities. We use $R^P$ to denote the paths (at least of length two), that connects two $EN$. A set of constraints $CT$ is given, with each $CT_i$ can be either instance level or concept level depending on its type, which will be given later. A tuple $t$ is a pair of two instances $(en_1, en_2)$, where $en_1 \in EN_i, en_2 \in EN_j$. A pattern $p$ consists of five parts:

(order, left, middle, right, $EN_1, EN_2$), where $EN_1, EN_2 \in EN$ and the order is a boolean variable denoting the order of the two entity types. Left, middle and right are word vectors $(w_i, tf_i)$ appearing within a fix window range $K_{\text{width}}$ on the left, middle and right position of the two entities during extraction. $w_i$ is the word appeared, and $tf_i$ is the regularized term frequency of the word.

To measure the similarity of two patterns, we will have a function $\text{sim}(p_i, p_j) : [0,1]^n \times [0,1]^n \mapsto [0,1]$. Since we can view each pattern as vectors of regularized word frequency, it is natural to apply cosine similarity to model the similarity function. Or alternatively, we can use string edit distance as our
We are given a small set of tuples $T^k_0$ for each $R_k$ as our input, and the extracted set of tuples iteratively grows as more tuples are extracted. Our evaluation metrics are precision and recall. We will introduce more symbols as we go deep into the model.

2.2 General Constraint Types

We give three general types of constraints according to our scenario.

**Mutual Exclusion Constraint** is a type of *Same Instance Different Concept Constraint*: $\forall t \in T, t \in R_i \implies t \notin R_j$. In our scenario, we take the StarIn relation and DirectedBy relation for every tuple of (Person, Movie), assuming that a person cannot appear as both actor and director in the same movie.

**Cardinality Constraint** is a type of *Different Instance Same Concept Constraint*: $\exists T_k \subset T, (\forall t_i, t_j \in T_k, t_i \in R \implies t_j \notin R) \land (\exists! t \in T_k, t \in R)$. In our scenario, we take the BirthYear of a person with such constraint, since for every person, he can only have one correct birth year.

**Evidence-Path Agreement Constraint** is a type of *Same Instance Same Concept Constraint (yet different view)*: $\forall t_i \in T_i, \forall t_j \in T_j, t_i = t_j \implies (t_i \in R \iff t_j \in R)$. In our scenario, we take the Actor-Movie-Actor path with StarIn relation, assuming that joining tuples along the path will get the same tuple as the direct extraction.

We have a general framework for solving all the above constraints. So before we look into each detailed model, we will introduce the big picture first.

2.3 Iterative Bootstrapping

As shown in figure 1.2, we utilize the idea of bootstrapping as weak supervision. Our whole co-extraction framework is based on the following algorithm. Here
extractPattern will go through the corpus and look for occurrences of each of the tuple and construct the pattern using the context appearing with the tuple; extractTuple will also go through the corpus, but look for tuples that appear with the context which match the top ranked patterns.

The main part of our model is on how to implement the Rank function. Typically, we will discuss this in the following.

\begin{algorithm}
\begin{center}
\textbf{Input:} $T^0_1, T^0_2, \ldots, T^0_n$, initial bootstrapping seed tuples for each relation; $D$, text corpus; $CT$, a set of constraints; $G^S$, schema graph; $K_p$, the number of top ranked pattern used for extraction; $K_t$, the number of top ranked extracted tuples.

\textbf{Output:} $T^k_1, T^k_2, \ldots, T^k_n$, the extracted tuples for each relation after $k$ iterations

\textbf{begin}
\begin{algorithmic}
  \State \textbf{for} $i = 1$ \textbf{to} $n$ \textbf{do}
  \State \hspace{1em} $P^0_i \leftarrow \emptyset$
  \EndFor
  \For{$k = 1$ \textbf{to} $K_{\text{MaxI}}$}
  \For{each relation $i$}
  \State $P^k_i \leftarrow \text{extractPattern}(D, T^{k-1}_i)$
  \EndFor
  \State $\text{Rank}(T^{k-1}, P^k, CT, G^S)$
  \For{each relation $i$}
  \State $T^k_i \leftarrow \text{extractTuple}(D, P^k_i, K_p, K_t)$
  \EndFor
  \EndFor
\end{algorithmic}
\textbf{end}
\end{center}
\end{algorithm}

\textbf{Algorithm 1: Iterative Bootstrapping Algorithm}

2.4 Ranking Model: TCP Graph

We followed the intuition from PRDualRank\cite{2} that the precision and recall of a tuple can be expressed by its corresponding patterns’ precision and recall in a tripartite inference graph. This inference graph $G^l = (T, C, P)$ should contain three parts: tuple, context and pattern. A context is a pair of tuple and pattern $(t, p)$, and one can think of a context as the snippet extracted.
The context can serve as the bridge between a tuple and a pattern. For example, if we have a pattern \( p: (\# \text{city is the capital city of} \ # \text{country}) \) and a tuple \( t: (\text{Washington D.C.}, \ \text{U.S.A.}) \), then the corresponding context can be achieved by plugging in the tuple \( t \) into pattern \( p \), i.e. (Washington D.C. is the capital city of U.S.A.). Examples of tuples, contexts and patterns are listed in figure 5.1.

![Figure 2.1: A high level conceptual tripartite inference graph. Tuples and patterns may belong to different relation types.](image)

We assume we got the ground truth of context for each relation \( i \), denoted as \( C_{R_i} \). Within \( C_{R_i} \) are the contexts that are truly denoting the relation of interest. Without ambiguity, we just use \( C_R \) to denote the ground truth when we are only considering one relation. We further define the sample space of a tuple \( t \) as \( I_t \), and the sample space of a pattern \( p \) as \( I_p \). The sample space is a set of contexts related to a tuple or pattern, that can be used to inference probability from. Thus, the precision and recall as for relation \( R \) is defined as:

\[
P(t) = Pr(c \in C_R | c \in I_t) \quad R(t) = Pr(c \in I_t | c \in C_R)
\]

\[
P(p) = Pr(c \in C_R | c \in I_p) \quad R(p) = Pr(c \in I_p | c \in C_R)
\]

There will be edges connecting the context with the extracted tuple and
pattern from that context, with edge weight denoting the number of occurrences of that context. Without further information (e.g. constraints), the edge only exists within the same relation type. Which means, the graph is reducible (separated), since there is no edge connecting tuples and patterns from different relation types. The following section will mainly be discussing how to utilize constraints to make this inference graph connected between different relation types (i.e. define $I_t$ and $I_p$), and how such connection can be expressed in terms of precision and recall.
Chapter 3

Inference with Mutual Exclusion Constraint

With an overview of the model above, we now give the details of the model for each constraint types as we have stated before.

We take the example constraint of Costar and DirectedBy relation, which are two mutually exclusive relationship for a singe tuple of (Movie, Person) assuming that a person cannot be both an actor and director of a single movie. We are aware that this assumption may not be true for some cases, but this is just an assumption in our scenario. One may consider other relations like Friends and Enemy relation between two person, where this assumption holds for every tuple.

Also, notice that this kind of mutual exclusive relationships existing within a single tuple is not just happen by chance. This is because ambiguity exists when the entity type is not fine-grained enough for the target relation [23, 24]. For example, if we have a entity tagger which can output with type Actor, then given a tuple (movie, actor), the relationship is much more clear than before. Here, we assume we don’t have such fine-grained entity tagger and such mutually exclusive relations are easy to find by substituting the entity type for some fine-grained types (e.g. Advisor and Classmate between (Person, Person) by substitute Person for Professor or Student).

Suppose the context that co-appear with a tuple t is denoted as $C_t$, and the context that co-appear with a pattern p as $C_p$. Since this mutual exclusive relationships is already embedded within a tuple, we define $I_t$ and $I_p$ as following:

$$I_t = C_t, I_p = C_p$$
Following the same inference derivation in PRDualRank, we get the following inference equations:

\[
P(p) = \sum_{t_i \in \pi(p)} P(t_i) \cdot \frac{|I_{tp_i}|}{|I_p|}, \quad R(p) = \sum_{t_i \in \pi(p)} R(t_i) \cdot \frac{|I_{tp_i}|}{|I_t|},
\]

\[
P(t) = \sum_{p_i \in \pi(t)} P(p_i) \cdot \frac{|I_{tp_i}|}{|I_t|}, \quad R(t) = \sum_{p_i \in \pi(t)} R(p_i) \cdot \frac{|I_{tp_i}|}{|I_{p_i}|},
\]

(if t is not seed)

Thus when sampling contexts from \( I_t \), we will be able to get contexts that denote Costar and DirectedBy. To specify the mutual exclusion, we use the labels from one against the other in the following algorithm. Intuitively, when inferencing for Costar, we treat all the DirectedBy input tuples as negative instances, and vice versa for DirectedBy inference.

| Input: \( T_1, T_2, \ldots, T_m \), seed tuples for each mutually exclusive relation; \( P_1, P_2, \ldots, P_m \), patterns for each corresponding relation; \( G^I = (T,C,P) \), inference graph |
| Output: Precision and recall for each tuple and pattern |
| begin |
| for each relation \( i \) do |
| begin |
| for each tuple \( t_k \in T \) do |
| if \( t_k \in T_i \) then |
| \( P(t_k) \leftarrow 1, R(t_k) \leftarrow \frac{1}{|T_i|} \); |
| end |
| if \( t_j \in T_j \& \& j \neq i \) then |
| \( P(t_k) \leftarrow 0, R(t_k) \leftarrow 0 \); |
| end |
| end |
| Inference Precision and recall of tuples and patterns for \( R_i \) in \( G^I \) until convergence. |
| end |
| end |

**Algorithm 2: Mutual Exclusive Inference**
Chapter 4
Inference With Cardinality Constraint

In our scenario, we consider the relationship BirthYear (Person, Year) to be a cardinality constraint: for each person, we think there is must be a unique correct tuple as his birth year. For the uniqueness to hold in practice, we will do a sampling for the tuples of each person before building up the TCP graph. To be specific, we sample for each person for all the years that related to him in the corpus. We assume such sampling is perfect: we will be able to include the correct tuple in the tuple base. And such assumption is practical for huge dataset like the web.

4.1 Basic Guiding Principles

Before starting to define our model, we want to first introduce the principles and assumptions behind our model. These are the starting point of all the derivation and one can see finally that our model will come to embrace the following principles.

**Sum of precision of tuples in the bag** Given a bag $B$, the sum of precision of all the tuples in the bag will equal to one. That is: $\sum_{t \in B} P(t) = 1$

Intuitively, this property is what we wanted as the Cardinality Constraint: one tuple getting high precision will lower the precision of the others, and vice versa since there must be a true tuple within a bag.

**Expansion of sample space for a tuple** With the cardinality constraint, we expand the sample space of a tuple to its bag’s covering contexts. That is: $I_t = I_{B(t)}$, where $B(t)$ denotes the bag that $t$ takes.
With the cardinality constraint, we expand the sample space of a single tuple \((I_t)\) to its whole bag while keeping \(I_p\) as the \(C_p\). Suppose \(T\) is the set of all the tuples, we can divide \(T\) into subsets \(\{T_1, T_2, T_3, \ldots, T_n\}\), with each subset denoting a bag. For example, for the relationship BirthYear, we have a bag for each person (e.g. \((Leonardo DiCaprio, \ast)\), \((James Cameron, \ast)\)). We use \(B(t)\) to denote the bag that \(t\) takes.

This principle intuitively holds due to the constraints that the tuples had within a bag: if a context denotes that the single tuple is true, one can infer based on the constraint that this tuple is true while the other tuples in the bag are false (because there can only be one correct tuple); if a context denotes that the single tuple is false, one can also infer similarly that this tuple is false while the other tuples in the bag have equal probability of being true (because there must be one unique correct tuple).

We further expand the notion of ground truth to make every tuple having it’s own ground truth, denoted as \(C'_{tR}\) for tuple \(t\). Note that this is consistent with the PRDualRank’s notion since we can simply take \(C'_{tR} = C_R\) for every tuple \(t\). The ground truth is nothing but the label set of a tuple, and towards the end, one can see that we are able to derive different label set for different tuples.

With the above principles and assumptions, we are able to derive the following:

\[
\sum_{t \in B} Pr(c \in C'_{tR}|c \in I_B) = 1
\]

\[
\Rightarrow \text{given } c \in B, \sum_{t \in B} Pr(c \in C'_{tR}) = 1(\ast)
\]

With (\ast), we are able to transfer the constraint that’s set on the tuples to contexts, and we need to derive \(C'_{tR}\) in order to satisfy this kind of property.
4.2 Extending Label Set

Ideally, we can have an oracle to denote the labeling of a context $C'_{R'}$ for us. One can think this oracle as a multiple-label oracle, which differs from the original one. For the sake of distinguishability, we use $c'$ for the contexts with the new label sets, as compared with $c$. Thus, for each context $c'$, and $B(c')$ being the bag the context belong to, the oracle can tell $\exists t \in B(c'), c' \in C'_{R'}$. However, in order to be consistent with PRDualRank as an extension, we would still assume that the oracle is the same: it can only denote whether $c' \in C_R$ or not, which is $C'_{R}^{T_e(c')}$ in our new setting, with $T_e$ denoting the tuple that originally extracted from the context.

To actually derive the new label set, we see there are two situations in the original label space: 1) when $c \in C_R$ and 2) when $c \notin C_R$. For 1), we know that in the new label space, $c'$ must be in $C'_{R}^{T_e(c')}$, and according to the constraint (*), we know that $\forall t \neq T_e(c'), c' \notin C'_{R}^{t}$. For example, if the oracle tells us the context "Leonardo, born in 1974, ..." is a true context of relation BirthYear, this context should be denoting that tuple (Leonardo, 1974) as true, while denoting other tuples like (Leonardo, 1997) as false. For 2), the only thing we know is that $c'$ must NOT be in $C'_{R}^{T_e(c')}$ for the rest of the tuples, we know there must be one tuple $t$ that $c' \in C'_{R}^{t}$ according to the constraint, but we don’t know which one. We utilize the Maximum Entropy Principle to infer the correct one: the rest of the tuples will have equal probability of being true. For example, from the fact that context "Leonardo, starred in ... 1997..." is a wrong context, we can infer that (Leonardo, 1997) is a wrong tuple, and the rest like (Leonardo, 1974), may get equal probability of being true. Without ambiguity, we use $B(c')$ to denote $B(T_e(c'))$ for simplicity and give the formal
derivation of $C'_t R$ from $C_R$ as:

$$Pr(c' \in C'_t R) = \begin{cases} Pr(c \in C_R) & T_e(c') = t_i \\ \frac{1}{|B(t_i)|-1}(1 - Pr(c \in C_R)) & T_e(c') \neq t_i \\ 0 & \land B(c') = B(t_i) \end{cases}$$

From the above definition, it is very easy to get the following equation, $\forall t_j \neq T_e(c') \land B(t_j) = B(c')$

$$Pr(c' \in C'_t t_j R) = (1 - Pr(c' \in C'^T_{t_e}(c'))) \cdot \frac{1}{|B(c')| - 1}$$

We give the example in figure 4.1 to show such derivation.

Figure 4.1: In the bag of three tuples, there are 12 contexts. The table for $c$ is formatted with $(contextName: ExtractedOccurrences)$; the table for $C_R$ is formatted with $(contextName: occurrences)$; the table for $C'_R$ is formatted with $(contextName: ExpectedOccurrences)$.

For example, for $c_1$, it is appearing 4 times in the original probability space.
The oracle in the original probability space tells us that this context is true. According to the derivation, this means \( c'_1 \) is in the label set of \( C'^{t_1}_R \), while not in \( C'^{t_2}_R \) or \( C'^{t_3}_R \). For \( c_4 \), the oracle tells us it is false, thus \( c'_4 \) must not be in \( C'^{t_1}_R \), and having equal probability of being in \( C'^{t_2}_R \) and \( C'^{t_3}_R \). We take expected count and that gives us 1 for each. Thus \( \Pr(c'_4 \in C'^{t_2}_R) = \Pr(c'_4 \in C'^{t_3}_R) = \frac{1}{2} \), which still sum up to 1.

We define the precision of tuple and pattern as:

\[
P(t) = \Pr(c' \in C'^t_R \mid c' \in I_t) 
\]

\[
P(p) = \Pr(c' \in C'^{T_\epsilon(c')}_R \mid c' \in I_p) 
\]

We give the following inference for bridging the context to tuple:

\[
\Pr(c' \in C'^t_R) = \sum_{t_i \in T} \Pr(c' \in C'^t_R, c' \in I_{t_i}) \tag{1}
\]

\[
= \sum_{t_i \in B(t)} \Pr(c' \in C'^t_R \mid c' \in I_{t_i}) \cdot \Pr(c' \in I_{t_i}) \tag{2}
\]

\[
= |B(t)| \cdot \Pr(c' \in C'^t_R \mid c' \in I_t) \cdot \frac{1}{|B(t)|} \tag{3}
\]

\[
= P(t) \tag{5}
\]

From (1) to (2), we enumerate that \( c' \) belong to all the \( t_i \). From (2) to (3), we omit the case that \( c' \) not belong to \( I_{t_i} \), since the probability will always be 0.

From (3) to (5), since for a single context \( c' \), every tuple in the bag is getting equal occurrence from it. Furthermore, \( \forall t_i \neq t_j \in B(t), I_{t_i} = I_{t_j} = I_t \), we simply use \( I_t \) for all the tuples and this will give us P(t). We give the similar bridging from context to pattern with two complementary conditions.

\[
\Pr(c' \in C'^t_R) \text{ if } T_\epsilon(c') = t \tag{1}
\]

\[
= \sum_{p_i} \Pr(c' \in C'^t_R, c' \in I_{p_i}) \cdot \Pr(c' \in I_{p_i}) \tag{2}
\]

\[
= \Pr(c' \in C'^t_R, c' \in I_{c',p}) \tag{3}
\]

\[
= \Pr(c' \in C'^{T_\epsilon(c')}_R \mid c' \in I_{c',p}) \tag{4}
\]

\[
= P(c',p) \tag{5}
\]
From (1) to (2) we sum over all the possible $p_i$ and use Bayes’ Rule for factorization. From (2) to (3), we use that fact that $c’$ has only one linking pattern, which we simply call it $c’.p$. From (3) to (4) we apply the condition that $T_e(c’) = t$. Similarly:

$$Pr(c’ \in C’_R) \text{ if } T_e(c’) \neq t \land B(c’) = B(t) \quad (1)$$

$$= (1 - Pr(c’ \in C’_{T_e(c’)})) \cdot \frac{1}{|B(t)|-1} \quad (2)$$

$$= (1 - P(c’.p)) \cdot \frac{1}{|B(t)|-1} \quad (3)$$

From (1) to (2) we use the equation derived before. Note that if $B(c’) \neq B(t)$, the probability will always be 0. From (2) to (3) we apply the results before.
4.3 Precision Inference Derivation

With the above properties, we are now able to give the following derivation of precision for a single tuple.

\[
P(t) = \Pr(c' \in C'_R | c' \in I_t) \quad \quad \text{(4.1)}
\]

\[
= \sum_{t_i \in B(t)} \Pr(c' \in C'_R | T_e(c') = t_i) \cdot \Pr(T_e(c') = t_i | c' \in I_{B(t)}) \quad \quad \text{(4.2)}
\]

\[
= \Pr(c' \in C'_R | T_e(c') = t) \cdot \frac{|C_t|}{|I_{B(t)}|} + \\
\sum_{t_i \in B(t) - \{t\}} \Pr(c' \in C'_R | T_e(c') = t_i) \cdot \frac{|C_{t_i}|}{|I_{B(t)}|} \quad \quad \text{(4.3)}
\]

\[
= \sum_{c'_m \in C_t} \sum_{t_i \in B(t) - \{t\} \cap c'_m \in C_{t_i}} \Pr(c' \in C'_R | T_e(c') = t_i) \cdot \frac{|C_{t_i}|}{|I_{B(t)}|} \quad \quad \text{(4.4)}
\]

\[
= \sum_{c'_m \in C_t} \sum_{t_i \in B(t) - \{t\} \cap c'_n \in C_{t_i}} \Pr(c' \in C'_R | T_e(c') = t) \cdot \frac{|c'_m|}{|C_{t_i}|} \frac{|C_t|}{|I_{B(t)}|} + \\
\sum_{t_i \in B(t) - \{t\} \cap c'_n \in C_{t_i}} \Pr(c' \in C'_R | T_e(c') = t_i) \cdot \frac{|c'_n|}{|C_{t_i}|} \frac{|C_t|}{|I_{B(t)}|} \quad \quad \text{(4.5)}
\]

\[
= \sum_{p_i \in \pi(t)} \frac{|I_{p_i}|}{|I_{B(t)}|} + \\
\sum_{t_i \in T - \{t\} \cap p_j \in \pi(t_i)} \frac{1}{|B(t)| - 1} \cdot (1 - \Pr(p_j)) \cdot \frac{|I_{t,p_j}|}{|I_{B(t)}|} \quad \quad \text{(4.6)}
\]

From (1) to (2) is based on our new definition of precision. (2) to (3) is using simple bayesian rules and enumerate different tuples in that bag. Note that \( I_t = I_{B(t)} \). (3) to (4) is picking out \( t \) from all the tuples. (4) to (6) is to enumerate over all the possible contexts in the bag and applying bayesian rules. (6) to (7) is using previous results. Similarly, we can get the precision
of a pattern, which is the same as the original one:

\[ P(p) = \sum_{t_i \in \tau(p)} P(t_i) \cdot \frac{|I_{pt_i}|}{|I_p|} \]

For recall, the cardinality constraint does not affect the recall of two tuple in the same bag. Intuitively, the concept that recall is considering should be consistent across all tuples (within or out of the bag), which should be consistent with the original \( C_R \). Thus the definition and derivation of recall should follow the previous ones.

\[ R(t) = Pr(c' \in T_e(t) | c' \in C^\prime_{R}(c')) = \sum_{p_i \in \pi(t)} R(p_i) \cdot \frac{|I_{pt_i}|}{|I_{p_i}|} \]

\[ R(p) = Pr(c' \in I_p | c' \in C^\prime_{R}(c')) = \sum_{t_i \in \tau(p)} R(t_i) \cdot \frac{|I_{tp_i}|}{|I_{t_i}|} \]
Chapter 5

Inference With Evidence-Path Agreement Constraint

Evidence-Path Agreement is another kind of constraint we would like to address in this paper. We will first introduce what is an evidence-path, and why such extraction should be considered, and then go into details of the model.

5.1 Evidence Path

We propose a certain kind of path – evidence path inside the schema graph $G^S$ that can serve as a guidance for the path-wise inference. We believe that the relations between any two entities are not just pair-wised, but also can be stated through interactions with other entities (often heterogeneous ones), thus path-wise. An evidence path is a path that if every relation defined on this path is correct, it will naturally imply that the target relation we want, from the starting entity to the ending entity of the path, is correct. We say $R^P(EN_i, EN_j)$ is an evidence path (which starts from entity type $EN_i$ and ends in $EN_j$), when

$$\forall en_i \in EN_i, en_j \in EN_j, t = (en_i, en_j) \text{ is true}$$

$$\iff \forall \text{Path}^t(en_i, en_j, R^P) \in R^P(en_i, en_j),$$

$$\forall t' = (en_a, en_b) \in \text{path}^t(en_i, en_j), \; t' \text{ is true}$$

where $\text{Path}^t(en_i, en_j, R^P)$ denotes the set of tuples joined by following the path $R^P$. For example, for $\text{Path}^t$(Leonardo, Kate, Actor - Movie - Actor) will include tuples (Leonardo, Titanic) and (Titanic, Kate). $R^P(en_i, en_j)$ denotes
the set of such paths. This gives us the definition of evidence path on the
schema graph. For our film domain schema graph, the evidence path we are
interested in is actor-movie-actor.

5.2 Different Ways To Extract

Having more ways to infer a relationship will definitely help for disambiguation,
since the semantic will become clearer with multiple explanations. However,
how to combine these expressions to help evaluate patterns and improve
precision and recall of the extraction is yet to be studied. Basically, we can
categorize the extraction of a tuple into two: direct extraction and path-wise
extraction. Direct extraction means to extract the target relationship by di-
rectly analyze the context where the tuple appear; Path-wise extraction means
to extract a tuple by joining several relationship tuples together. For example,
extracting the fact that A stars in a film B by joining “A starring as character
C” and “C being the character in movie B”.

Direct extraction typically just analyze the context where the tuple appears,
and the analysis is limited to a certain window frame or a single page. Thus,
single extraction cannot handle situations when the relationship is expressed
in separate tuples. For example, Leonardo and Kate both starred in the movie
Titanic, and thus they have the costar relationship. However, there may be
few or no contexts directly say that they costarred with each other before. But
rather, they would both appear with Titanic in lots of contexts. So extraction
along Leonardo-Titanic-Kate will be better than direct extraction. Path-wise
extraction, on the other hand, can be a good fit for such situations, but we
should be cautious when joining the tuples. The entity ambiguity (e.g. two
films with the same name) will render our joining result as totally wrong. Also,
since this kind of path will be applied to the joining of other similar tuples,
we need to evaluate in the schema level which path should be joined.
5.3 Relation Expressed In Path-wise Way

A context is the combination of a tuple and a pattern. Following the evidence paths defined in the schema graph, we can derive the path-wise context for a specific tuple. Note that here the context is different from the original context because it can combine several contexts together as the path-wise context for that tuple. Given the extracted context, which can be constructed with extracted tuples and patterns, the derived contexts of relation 1 (CoStar between actor and actor) for tuple are given in figure 5.1.

From figure 5.1 we know that the derived context for relation 1 should be: $C^1 = \{c_1 : 4, c_2 : 2, (c_3, c_5) : 8, (c_4, c_5) : 12, (c_3, c_6) : 2, (c_4, c_6) : 3, (c_5, c_6) : 4\}$.

If we have an oracle to determine the ground truth of relation 1, then we can give the ground truth as: $C^1_R = \{c_1 : 4, c_2 : 2, (c_3, c_5) : 8, (c_4, c_5) : 12\}$. With an oracle, we can easily judge that $(c_3, c_6) : 2$, which is extracted along (Leonardo Dicaprio - Titanic - Daniel Radcliffe), is a wrong extraction. For any simple context $c$ (non-composite), we can still view that $c$ only link with one tuple and one pattern, so the original equation in PRDualRank still holds:

$$P(c) = P(c.t) = P(c.p)$$  \hspace{1cm} (5.1)

$$R(c) = R(c.t) = R(c.p)$$  \hspace{1cm} (5.2)

For composite contexts, we need to derive their probability from simple ones. For this sake, we give the following independence assumption for the rule of breaking down path-wise contexts into individual simple ones.

**Independence Assumption for evidence path** Given $c^1_{ki}, c^2_{ki}, \ldots, c^k_{ki}$ forming an acyclic path as the i-th instance of the k-th evidence path (any entity instance should not appear in the path for more than once), then $c_i$ is independent of each other. This means:

$$Pr(c_{ki} \in C_R) = \prod_j Pr(c^j_{ki} \in C^\text{type}(c^j_{ki}))$$

23
Intuitively, this assumption holds because in an evidence path, two types of relations only share one type of entity with and only with the neighboring types of relations. Suppose that \( m = \text{len}(c_{ki}) \), so it is easy to see that \( c_{ki}^j \) only depends on \( c_{ki}^{j-1}, j \in [2, m] \), with this property, we can rewrite the joint probability of \( c_{ki1}, c_{ki2}, \ldots, c_{kim} \) as:

\[
Pr(c_{ki} \in C_R) = Pr(c_{ki1} \in C_R^{\text{type}(c_{ki1})}, c_{ki2} \in C_R^{\text{type}(c_{ki2})}, \ldots, c_{kim} \in C_R^{\text{type}(c_{kim})}) \\
= Pr(c_{ki1} \in C_R^{\text{type}(c_{ki1})}) \cdot Pr(c_{ki2} \in C_R^{\text{type}(c_{ki2})}|c_{ki1} \in C_R^{\text{type}(c_{ki1})}) \cdot \ldots \cdot Pr(c_{kim} \in C_R^{\text{type}(c_{kim})}|c_{kim-1} \in C_R^{\text{type}(c_{kim-1})})
\]

Furthermore, we assume that the sharing of only one entity between two relations wouldn’t place any constrains on the conditional probability. For example, if an actor starred in a movie, the probability of this movie being directed by a specific director will not be narrow down under our independence assumption. So the result of the above equation should just be:

\[
\prod_j Pr(c_{ki}^j \in C_R^{\text{type}(c_{ki})})
\]

Here \( \text{type}(\cdot) \) is a function that maps to the specific type number of a relation. When the context is clear, we will ignore such mapping. Note that the first line is the definition of the evidence path, because the whole path being correct is equivalent to just the individual binary relations being correct. And this is important in this independence assumption since one may think of other definitions of the path’s probability which take the dependency between relations into account.

To actually incorporate the evidence path into the inference model, we need
to define the context evidence path of a tuple $t$ as:

$$Path^c(t) = \{ (c_1, c_2, \ldots, c_k) | c_i \in C_{t_k}, i \in [1, k], \text{ where } t_k \in Path^l(t) \}$$

There may be several evidence paths for a specific relation, then we further denote the $i$-th instance of evidence path $k$ as $c_{ki}$, and the $j$-th relation in $c_{ki}$ as $c^j_{ki}$. For example, the relation of CoStar(actor, actor), can have 3 evidence path: actor-actor, actor-movie-actor, actor-character-movie-actor, so $k \in [1, 3]$. Then, since there may be several contexts extracted for such a path (e.g. $c_1$, $c_2$ for path 1 (actor-actor) and $(c_3, c_5), (c_4, c_6), (c_5, c_6)$ for path 2 (director-movie-director)), we denote each context with a subscript $i$. (i.e. $c_{11} = c_1, c_{12} = c_2$, etc. $c_{21} = (c_3, c_5), c_{22} = (c_4, c_5)$, etc.). To further denote each binary context in a complex context, we use a superscript $j$. For context paths like $c_{21}$, we have $c^1_{21} = c_3, c^2_{21} = c_5$.

When extracting, we often only care about a specific tuple. For example, what are the contexts related to the tuple of (Leonardo DiCaprio, Kate Winslet)? Because the inference of different tuples should be separated from each other, we need a definition of fanout of a specific tuple to filter those irrelevant context. We then denote the fanout of a specific tuple $t$ as:

$$I_t = \{ c | c \in Path^c(t) \}$$

To find the fanout of $t$, we first need to find the contexts paths constraining on that tuple. For tuple (Leonardo DiCaprio, Kate Winslet), we have the following instances for each path. For path 1 (actor, director), we have $c_1$ and $c_2$; and for path 2 (actor–movie–director), we have $(c_3, c_5)$. After ordering them according to the derived context table in figure 5.1, we have, for example, $c_{11} = c_1, c_{12} = c_2, c_{21} = (c_3, c_5)$. For context paths like $c_{21}$, we have $c^1_{21} = c_3, c^2_{21} = c_5$.

Then, with these contexts, the fanout is just union of all the individual binary
contexts mentioned above.

\[ I_{(LeonardoDiCaprio,KateWinslet)} = \{c_1, c_2, (c_3, c_5), (c_4, c_5)\} \]

Compared with this, the fanout of a pattern should not include in the path-wise context. That is because a pattern is relation specific, which means it should only be connected to the context of its own relation. For example, \( I_{p_1} = \{c_1\} \). Following the intuition and definition above, a general inference model between tuples and patterns for the CoStar(actor, actor) relationship is shown in Figure 5.1. Note that there will be a similar general inference model for every different relation, since the evidence path for each relation, hence the inference paths between tuple and pattern, should be different. Inside this inference model, the tuples are instances of (actor, actor); the patterns come from different related individual patterns, i.e. from StarIn, and CoStar. The contexts are modified into path-wise contexts. Smaller pair-wised contexts can be combined to form path-wise contexts, and the path-wise contexts directed linked with the tuples are defined by evidence paths for that relation.

Specifically, for the tuple of (Leonardo DiCaprio, James Cameron), the left general inference graph can be particularly explained through the extracted facts listed in figure 5.1. The final inference graph of CoStar will couple with StarIn, since the evidence path is defined on StarIn. So one can imagine the inference graph by extending the context graph and thus the tuples and patterns related to both relations. Inside this graph, the red lines denotes how the inference will be made. For example, we know that \( I_{(Leonardo,Kate)} = \{c_1, c_2, (c_3, c_5), (c_4, c_5)\} \), so there will be edges from the tuple (Leonardo DiCaprio, Kate Winslet) to \( c_1, c_2, (c_3, c_5) \) and \( (c_4, c_5) \). \((c_3, c_5)\) will further distribute the inference to \( c_3 \) and \( c_5 \) individually and same for \( (c_4, c_5) \). Then from the context, we can inference to patterns by looking up which pattern the context is composed of. For example, \( p_1 \) composes \( c_1 \), so there will be inference edge from \( c_1 \) to \( p_1 \). And the same will be between \( c_3 \) and \( p_3 \).
5.4 Derivation Detail

With the definition above, we are now able to derive the precision of a specific tuple instance based on the evidence path. Let’s take the cooperation relation as our example. As definition, the precision should be:

\[ P(t) \]

\[ = Pr(c \in C_R | c \in I_t) \]  
\[ = \sum_{c_i \in C} Pr(c \in C_R, c = c_i | c \in I_t) \]  
\[ = \sum_{c_i \in I_t} Pr(c \in C_R | c = c_i, c \in I_t) \cdot Pr(c = c_i | c \in I_t) \]  
\[ = \sum_{k} \sum_{c_{ki} \in I_t} Pr(c \in C_R | c = c_{ki}, c \in I_t) \cdot Pr(c = c_{ki} | c \in I_t) \]  
\[ = \sum_{k} \sum_{c_{ki} \in I_t} Pr(c_{ki} \in C_R) \cdot \frac{|I_{c_{ki}}|}{|I_t|}, \]  

From (4) to (5),

\[ = \sum_{k} \sum_{p_{ki} \in \pi(t)} \prod_j P(p_{ki}^j) \cdot \frac{|I_{p_{ki}}|}{|I_t|}, \]

where \( |I_{c_{ki}}| = \prod_j |c_{ki}^j| \), where \( c_{ki}^j \in c_{ki}, j = 1, 2, \ldots \text{len}(c_{ki}) \)

\[ = \sum_{k} \sum_{c_{ki} \in I_t} \prod_{j} Pr(c_{ki}^j \in C^\text{type}(c_{ki}^j)) \cdot \frac{|I_{c_{ki}}|}{|I_t|}, \]

where \( \text{type}(c_{ki}^j) \) returns the relation type of \( c_{ki}^j \)

\[ = \sum_{k} \sum_{c_{ki} \in I_t} \prod_{j} P(c_{ki}^j) \cdot \frac{|I_{c_{ki}}|}{|I_t|} \]  
\[ = \sum_{k} \sum_{p_{ki} \in \pi(t)} \prod_{j} P(p_{ki}^j) \cdot \frac{|I_{p_{ki}}|}{|I_t|} \]
we add different types of evidence paths into the summation by breaking $c$ into different types of $c$, corresponding to different paths. In (6), we make the definition of $|I_{c_k}|$ like this because of the independence assumption between binary contexts stated below, since if the contexts are independent, we can count the possible combinations of contexts by multiplication. From (6) to (7), we use the independence assumption of evidence path contexts. From (7) to (8) and (8) to (9), since each $c_{ki}$ are just the simple pair-wise relation context, we utilize the derivation from PRDualRank to bridge from context to pattern. $I_{t_k p_k}$ is essentially equal to $I_{c_k}$, with only a change in notation.

5.5 Interpretation of Evidence Path Inference

Adding the evidence path as a way of inference for the tuple is analogous to the concept of External Classifier in semi-supervised learning [10]. Basically, for each tuple that have an evidence path for inference, the cross-relation inference score can be viewed as a external knowledge in judging the final precision and recall of that tuple. However, instead of manually assigning weights of $\eta$ and $1 - \eta$ to balance the inference between external and internal classifiers, we directly "learn" the weight from the proportion of extraction count (edge weight) inside the heterogeneous inference graph $G^I$. Our approach is more flexible as it denotes the favoring of the better extraction path with higher weight, which corresponds to the tradeoff between direct and path-wise extraction as we discussed before.
Figure 5.1: An example of extracted tuples and patterns. These facts can be constructed into contexts, where a context is simply a combination of a tuple and a pattern. The derived contexts are based on the evidence paths defined in the schema as well as the extracted contexts. The right graph is the inference graph for CoStar relationship, with derived contexts constructed from the left.
Chapter 6
Experiments and Empirical Study

In this section, we will show our experiment on relation extraction with constraints added. We take an example relation for each constraint and overall, our method outperforms the baseline significantly.

6.1 Basic Experiment Settings

**Dataset.** Our dataset is the Wikipedia dump[1]. The dataset consists of huge amount of english-language text pages. For more scalable experiment, we filtered out the pages that are not related to our domain (film) by checking the category keywords. This still leaves us with 234,100 pages. Moreover, we also filters out those structured data on Wikipedia (e.g. the info-box for each entity) since we are mainly dealing with natural language patterns, instead of structural patterns.

**Relations and constraints.** As shown in figure[1.1] we take 4 relation types: StarIn(Movie, Person) (R1), DirectedBy(Movie, Person) (R2), CoStar(Person, Person) (R3), BirthYear(Person, Year) and consider 3 types of constraints: Mutual Exclusion(R1, R2), Cardinality Constraint(R4), Path-view Constraint(R3, R1-R1).

**Ground truth generation.** We first get the ground truth from Freebase[2] using MQL API[3]. We further randomly pick 2000 films from the above and all the related information (actors, directors, birth year etc.). Then we take the

intersection of all the tuples from our dataset and the freebase golden tuples as our ground truth.

**Pattern and tuple extraction.** We first use Stanford NER tagger with 7 class (Time, Location, Organization, Person, Money, Percent, Date) to tag the dataset pages. For films, we use dictionary-based entity tagger from LingPipe. For tuple and entity extraction, we analyze the contexts within a certain window of $K_w$ words. When comparing two patterns for extraction, we set the similarity function as:

$$
sim(p_i, p_j) = ld(left_{i,j}) \times \alpha_1 + ld(middle_{i,j}) \times \alpha_2 + ld(right_{i,j}) \times \alpha_3$$

if $p_i.order = p_j.order$, otherwise 0. Function $ld(\cdot)$ is the Levenshtein distance (string edit distance) and $\alpha$ are parameters as weights for each part.

**Baseline 1.** We use the PRDualRank with iterative extraction as our first baseline. This is a state-of-art algorithm which ranks patterns and tuples simultaneously by exploring the duality between them. However, the model fails to address constraints within and between relationships. Here, we use the baseline to extract tuples from each relationship without enforcing any constraints. To be fair, we run the baseline and our model on the same dataset, with the same input as initial labels.

**Baseline 2.** We use Couple Semi-supervised Learning as our second baseline. Their proposal is similar to ours because they are enforcing the constraints in the way they combine the relations together. However, they lack a systematic way of combining evaluation of patterns with constraints. So by comparing with this baseline, we can see the significance of our ranking model as discussed above.

**Evaluation Metrics.** We measure the precision and recall of extracted tuples against the ground truth tuples. We provide the f1 curve for the precision and recall of our model comparing with the baseline.

**Parameters** As seen before, we have several parameters to tuned. In ex-

\[^{4}\text{http://nlp.stanford.edu/software/CRF-NER.shtml}\]
\[^{5}\text{http://alias-i.com/lingpipe/demos/tutorial/ne/read-me.html}\]
## Table 6.1: Statistics of dataset.

<table>
<thead>
<tr>
<th>Relation</th>
<th>Description</th>
<th>Constraint Type</th>
<th>#All Tuples</th>
<th>#Correct Tuples</th>
<th>#Initial Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>StarIn</td>
<td>A person act in a movie</td>
<td>Mutual Exclusion</td>
<td>6702</td>
<td>2318</td>
<td>20</td>
</tr>
<tr>
<td>DirectedBy</td>
<td>A person directed a movie</td>
<td>Mutual Exclusion</td>
<td>6702</td>
<td>1049</td>
<td>20</td>
</tr>
<tr>
<td>BirthYear</td>
<td>A person born in a specific year</td>
<td>Cardinality</td>
<td>34,203</td>
<td>5329</td>
<td>50</td>
</tr>
<tr>
<td>Costar</td>
<td>A person act with another person in a movie</td>
<td>Path View</td>
<td>90,807</td>
<td>58507</td>
<td>100</td>
</tr>
</tbody>
</table>

## Table 6.2: Experiment settings and statistics.

Experiment, we set our parameters as in table 6.3, note that the relations are numbered according to figure 1.1

6.2 Overall Performance

We provide the result of each relation extraction task with constraints in figure 6.1. Apparently, our model outperforms the baseline significantly. After examining the details of the inference process, we found that:

1. Our model combine the constraints with the pattern ranking and selection process. For example, we can see that with mutual exclusion constraint, we rank patterns like "...<person>stars in <movie>..." high while "...<movie>by <person>..." as low. Furthermore, we get lower ranking for general patterns like "...<movie>, <person>..." than the baseline because it can extract two competing semantics. This is differ-
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_w$</td>
<td>words in the contexts</td>
<td>20</td>
</tr>
<tr>
<td>$K_i$</td>
<td># initial Input for each relation</td>
<td>(20,20,50,100)</td>
</tr>
<tr>
<td>$K_e$</td>
<td># tuples extracted for each iteration</td>
<td>(10,10,100,500)</td>
</tr>
<tr>
<td>$K_p$</td>
<td># patterns used for each extraction</td>
<td>(30,30,80,100)</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>weight for comparing patterns</td>
<td>(0.25,0.5,0.25)</td>
</tr>
<tr>
<td>$K_{sim}$</td>
<td>similarity threshold for comparing patterns</td>
<td>0.7</td>
</tr>
<tr>
<td>$K_{MaxI}$</td>
<td>Maximum times of iteration for inference</td>
<td>1000</td>
</tr>
</tbody>
</table>

Table 6.3: Table of parameters setting.

ent from previous works where they just filtered out tuples that breaks such constraint.

2. Our model still works when semantic drift is severe. For relationship of BirthYear, the semantic is easily drifted to other general semantics like "ImporantYear". However, our model can still reduce the rate of semantic drift in BirthYear, with the bag-based constraint.

3. Our model can well combine path-wise patterns and directed patterns, and thus explore two different view of extraction at the same time. For example, tuples with direct patterns like "<person> with <person>" may get a decent score in direct extraction, but we can differentiate the tuples from the path-wise way: if extracted by "...<movie>starring <person>.." and "...<person>stars in <movie>...", the score will be boosted by the path-wise pattern score; if extracted by "...<movie>starring <person>.." and "...<movie>directed by <person>...", the score will be damped.
6.3 Discussion

It is worth noting that different relationship will have different difficulty in extraction. The difficulty can be determined by: 1) the selectivity of the relationship, which can in turn be intuitively determined by the number of correct tuples against the number of all tuples. For example, for BirthYear relationship, we have only 5329 out of 34203 tuples are correct, which means we need very specific patterns to pick out the tuples. 2) the rate of semantic drift. (e.g. BirthYear can be easily drifted to other relations like "important year" of that person.) Thus, the performance of each relation should be different based on the difficulty factor. But in spite of such factor, our model can still work on each relationship.

It is also worth noting that, our model, though can achieve higher f1 score than the baseline, we cannot guarantee the elimination of semantic drift. That is to say, even with constraints and better performance, semantic drift can
still occur since this is a known issue with bootstrapping method and to truly eliminate that we need more labels or guidance along the iterative extraction.

6.4 Conclusion and Future Work

With all the contents above, we have presented a model that can extract relation tuples by bootstrapping, while evaluating the patterns in a constraint-based metric-aware way. The experiment on Wikipedia dump showed that our model can increase the f1 score of our extraction result by a large margin as compared with two state-of-art baselines. Since PRDualRank only addresses metric-aware evaluation, and coupled constraint learning models only addresses constraints, it is proven that our good result is coming from the combination of meta-level constraints and metric-aware evaluation.

We do see several extension point of our work. First, we can extend our setting to OpenIE. OpenIE has the advantage of automatically extracting lots of different relation types and up till now we still haven’t seen any related work on how to couple the relations together. We believe if we can specify our constraints in a more generic way to fit in OpenIE’s setting, we will be able to develop similar models as well. Second, we haven’t consider the scalability issue as the extraction tuple number increases. The TCP graph may not even fit into main memory. There should be a way to develop efficient and scalable algorithms for inferencing on TCP graph.
References


[13] C. C. A. Yizhou Sun, Jiawei Han and N. V. Chawla, “When will it happen? — relationship prediction in heterogeneous information networks,” WSDM, 2012.


