# Ontological Smoothing for Relation Extraction

Congle Zhang, Raphael Hoffmann, Daniel S. Weld
Computer Science & Engineering
University of Washington
Seattle, WA 98195, USA
{clzhang,raphaelh,weld}@cs.washington.edu

## ABSTRACT

There is increasing interest in *relation extraction*, methods that convert natural language text into structured knowledge. The most successful techniques use supervised machine learning to generate extractors from sentences which have been labeled with the arguments of the relations of interest. Unfortunately, these methods require hundreds or thousands of training examples, which are expensive and time-consuming to produce.

This paper presents *ontological smoothing*, a semi-supervised technique that exploits knowledge of a background ontology in order to learn extractors for a set of minimally-labeled relations. Ontological smoothing has three phases. The first step generates a mapping between the target relations and the background ontology. The second step uses knowledge-based weak supervision to heuristically generate new training examples for the target relations. The third step learns an extractor from a combination of the original and newly generated examples. Experiments on 61 relations across two target domains with Freebase as the background ontology show ontological smoothing can dramatically improve precision and recall.

## General Terms

Relation Extraction

## 1. INTRODUCTION

Vast quantities of information are encoded on the Web in natural language. In order to render this information into structured form for easy analysis, researchers have developed methods for *relation extraction*, also known as information extraction. The most successful techniques use supervised machine learning to generate extractors from a training corpus comprised of sentences which have been labeled with the arguments of the relations of interest. For example, the sentence " 'Our captain, Jelani Jenkins, is a phenomenal athlete' said the Gator's head coach, Urban Meyer." might have "Jelani Jenkis" and "Urban Meyer" annotated as arguments for the `isCoachedBy` relation. Unfortunately, these supervised methods require hundreds or thousands of training examples per relation, and thus have proven too expensive for use in constructing Web-scale knowledge bases.



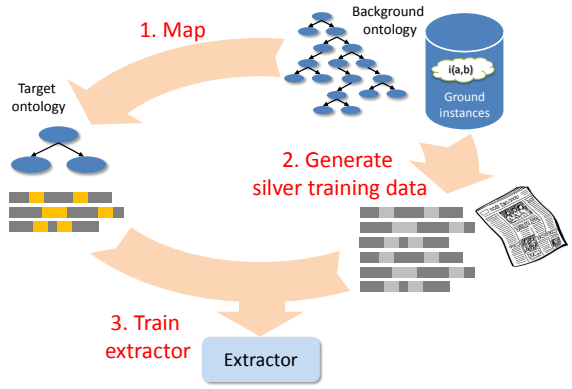



Figure 1: VELVET operates in three phases. First, it generates a global mapping from view on the background ontology onto the target relations. Second, it uses the mapping to create silver labeled training data for the target relations. Finally, it trains relation extractors for the targets.

In the absence of copious training data, effective learning is thought to require some form of additional constraint or knowledge on the part of the learner. But only a few methods have been proposed for leveraging background knowledge in order to improve learning.

This paper presents VELVET, a new, semi-supervised technique, called *ontological smoothing*, that learns extractors from a set of minimally-labeled relations by exploiting knowledge encoded in a background ontology and a large, unlabeled textual corpus. VELVET works in three phases as Figure 1 shows: the first step generates a mapping between the relations in the target ontology and the background ontology. The second step uses knowledge-based weak supervision to heuristically generate new, silver training examples for the target relations[1]. Finally, the third step learns an extractor from a combination of the original and newly-generated, candidate examples.

Mapping between ontologies is more complex than simply choosing a background relation which seems most similar to the target relation. In order to be able to find a quality mapping for a large number of target relations, one must consider the much larger space of mappings formed by unions of conjunctive queries on the background ontologies. In the related problem of database schema integration, these correspond to *views* defined using SQL operations select, project, join and union. Even if one limits the search to expressions using no more than two joins over three background rela-

[1]We use the term 'silver' to denote heuristically generated training data, which likely contains noise and is not as valuable as gold-standard, labeled data.

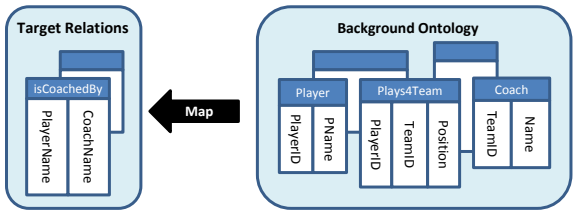



Figure 2: In order to map a target relation to the background ontology, one must consider a large space of possible views. In this example, the target `isCoachedBy` relation maps the following expression over Freebase relations: $\pi_{\text{PName,Name}}$Players $\bowtie$ Plays4Team $\bowtie$ fbCoach. In fact, the best mapping is a union of this expression with similar expressions for Freebase football, baseball and basketball relations.

tions, the search space may be huge, comprising billions of possible views.

## 1.1 A Simple Example

For example, suppose that we wish to learn extractors for the target relations defined by the Nell ontology as described in [5] and suppose that we use Freebase [4] as the background ontology. Consider target relation `isCoachedBy` whose instances are entity pairs of athletes and their coaches (Figure 2). While Freebase has considerable background information about athletics, it is broken down into separate relations for individual sports and has been normalized in a manner that precludes simple mapping to `isCoachedBy`. In order to find an accurate mapping, one needs to consider multi-join views, defined using database queries, and furthermore one must choose the best possible view from the myriad available. The mapping in our VELVET implementation uses Markov Logic and blocked Gibbs sampling to perform fast, approximate, probabilistic inference during the mapping process, and returns the following expression as the best map for this example[2]:

**Example 1**

$$\pi_{1.\text{name},2.\text{name}}(\text{baseballPlayer}^1 \bowtie \text{baseballCurrentTeam} \bowtie \text{baseballRosterPosTeam} \bowtie \text{baseballManager} \bowtie \text{coach}^2)$$
$$\cup\ \pi_{3.\text{name},4.\text{name}}(\text{footballPlayer}^3 \bowtie \text{footballTeam} \bowtie \text{footballRosterPosTeam} \bowtie \text{footballTeamHeadCoach} \bowtie \text{footballCoach}^4)$$
$$\cup\ \pi_{5.\text{name},6.\text{name}}(\text{basketballPlayer}^5 \bowtie \text{drafted} \bowtie \text{draftedTeam} \bowtie \text{basketballTeamCoach} \bowtie \text{coach}^6)$$

## 1.2 Smoothing with KB Weak Supervision

Once the system has found a mapping from the background ontology, VELVET can generate new candidate tuples for the target relations. Continuing the example, suppose that when the view is executed on Freebase it returns the fact that "Will Muschamp" is the coach of "Jeff Demps." VELVET searches an unlabeled corpus of text, such as New York Times articles, for sentences containing synonyms for both entities. Knowledge-based weak supervision treats each such heuristically-labeled sentence as a positive example of the `isCoachedBy` relation [18, 16], and VELVET learns a CRF extractor with a combination of the original and newly-generated training data.

Compared to prior KB weak supervision works [30, 17, 16] focusing on atomic relations defined already in a large ontology (*e.g.* Freebase, Wikipedia), this paper is the first work to extend weak supervision to any relation defined in few training examples.

## 1.3 Contributions

The rest of this paper details the process of ontological smoothing, but the most attention is given to VELVET's method for generating the ontological mappings. Section 2 formulates the mapping problem as finding the highest probability global mapping between all target entities, types and relations and the background ontology. Next, section 3 explains how we compute the best mapping using Markov Logic and blocked Gibbs sampling to perform fast, approximate probabilistic inference. Section 4 summarizes how we use the mapping to generate new examples and perform knowledge-based weak supervision. In section 5 and 6 we describe the experimental setup and results. Section 7 discusses related work, and section 8 concludes. Overall, this paper makes the following contributions.

1. We introduce ontological smoothing, a novel approach for learning relation extractions with minimal supervision.

2. Our approach is based on a new ontology-mapping algorithm, which uses probabilistic joint inference on schema- and instance-level features to explore the space of complex mappings defined using SQL selection, projection, join and union operators.

3. We present experiments on two target ontologies, using Freebase as background knowledge, that demonstrate that ontological smoothing can produce dramatic improvements to both the precision and recall of extraction.

# 2. MAPPING BETWEEN ONTOLOGIES

The most important step in ontological smoothing is the construction of a mapping between the target and background ontologies. This subject has been studied extensively by several different communities, often using different terminology, *e.g.*, schema mapping [28] in data management and ontology mapping in the semantic web [25]. While the literature is too vast for a comprehensive survey, the next subsection provides the context for our work. Then, the remaining subsections describe our approach. Section 2.2 defines the class of mappings considered. Section 2.3 describes a heuristic filter used to reduce the size of the search space, and Section 2.4 introduces a probabilistic model which allows us to choose the best mapping.

## 2.1 Context and Previous Work

Dhamankar et al.[12] define schema *matching* to be the first step in the process of constructing a *mapping*, *i.e.* a function converting descriptions of objects in one ontology into corresponding descriptions in another. We consider ontologies comprised of *types* (unary relations, also known as concepts, organized in a taxonomy) and binary *relations*. Relations may connect two types (*e.g.*, *Parent*) or may link a type to a primitive value, such as numbers, dates and strings (*e.g.*, *BirthDate*), which are often called *attributes* or *properties*. Each type is associated with a set of instances, called *entities*.

[2]We abbreviated the names of the original Freebase relations for better readability, *e.g.* we use basketballPlayer instead of /basketball/basketballPlayer. We use superscript indices to provide a shorthand for relation names in projections. For example 1.name is shorthand for basketballPlayer.name as indicated by the superscript on the basketballPlayer relation in the join.

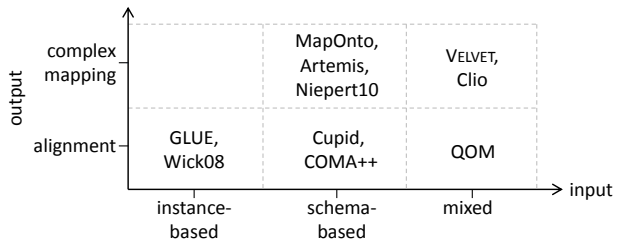



Figure 3: Classification of selected ontology matching systems, based on [15].

A *mapping* from a background ontology $B^{\mathcal{O}}$ onto a target $T^{\mathcal{O}}$ is a set of partial functions whose ranges are entities, types and relations in $T^{\mathcal{O}}$. Ullman [34] noted that these mappings can be thought of as view definitions, *e.g.* defined using SQL operations such as selection, projection, join and union. We adopt this perspective as shown in Example 1.

Euzenat & Shvaiko [15] and Rahm & Bernstein [28] carve the set of approaches for ontology matching into several dimensions. The input of the matching algorithm can be *schema-based*, *instance-based* or *mixed*. The output can be an *alignment* (*i.e.*, a one-to-one function between objects in the two ontologies) or a *complex mapping* (*e.g.*, defined as a view). Figure 3 plots some previous methods along these dimensions.

The majority of existing systems focus on the alignment problem. Doan *et al.* [13] present GLUE, which casts alignment of two taxonomies into classification and uses learning techniques. The more recent system by Wick & McCallum [36] applies a learning approach to a single probabilistic model that considers all matching decisions jointly. While these system operate on instances, others align schemas: Cupid [20] matches tree-structures in three phases, that include linguistic matching, structural matching, and aggregation. COMA++[2] enables parallel composition of matching algorithms. Niepert *et al.* [27] propose a joint probabilistic model based on Markov logic. QOM [14] matches both, instances and schemas, and is able to trade off between efficiency and quality.

Far less work has looked at finding complex mappings between ontologies. Artemis [8] creates global views using hierarchical clustering of database schema elements. MapOnto [1] produces mapping rules between two schemas expressed as Horn clauses. Miller *et al.*'s tool Clio [22][23] generates complex SQL queries as mappings, and ranks these by heuristics. Carman [7] can learn semantic descriptions of web information sources, it involves some complicated calculations for weather forecasts and travel deals.

For ontological smoothing to work, it is essential that one can find complex mappings involving selections, projections, joins, and unions. While MapOnto and Clio handle complex mappings, they are semi-automatic tools that depend on user guidance. In contrast, we designed VELVET to be fully autonomous. Unlike the other two, VELVET uses a propabilistic representation and performs joint inference to find the best mapping.

## 2.2 Ontologies and Views

We assume that an ontology is defined in terms of unary types, $t$, and binary relations, $r$. We denote the selectional preference (type constraint) of a relation by writing $r(t_1, t_2)$. For example, `isCoachedBy(athlete,coach)` is a relation in the Nell ontology [5]. We assume that each target relation comes with a set of seed instances $\{\ldots(e_1, e_2)\ldots\}$, where $(e_1, e_2)$ is an entity pair.

Typically the background ontology is comprised of many types and relations and is populated with numerous entities. There are also many relations defined on the entities. As we mentioned in the introduction, in order to be able to find a quality mapping for the target relation, one must consider the large space of mappings formed by the set of database *views* defined using the relational operations of select, project, join, and union. When mapping into a unary type of the target ontology we consider views comprised of unions over the background types.

When constructing a mapping to a target relation, it is often useful to select a subset of a background relation or view. One way that VELVET performs selection is by adding types into the join. Consider the following views:

**Example 2**

$$\pi_{1.name,2.name}\ \ location^1 \bowtie containedBy \bowtie location^2$$
$$\pi_{city.name,country.name}\ \ city \bowtie containedBy \bowtie country$$
$$\pi_{sportsFacility.name,city.name}\ \ sportsFacility \bowtie containedBy \bowtie city$$

Note that the second and the third views are a subset of the first and denote very different relations. It is not correct to use the instances of the first view to train the relation extractor for target relation *cityLocateInCountry*.

Formally, we represent an ontology mapping between target relation $\hat{r}(\hat{t}_1, \hat{t}_2)$, and the background view defined as $\cup r(t_1, t_2)$, where $r$ is created by joining the binary relations in the background ontology (e.g. basketballPlayer $\bowtie$ basketballRosterPosition $\bowtie$ basketballTeam $\bowtie_{\text{coach}}$ headCoach). $t_1$ and $t_2$ are selection operations defined by corresponding entity types. Without explicit explanation, we will from now on use $\hat{r}$, $\hat{t}$, $\hat{e}$ to denote target relations etc, and $r$, $t$, $e$ to denote ones in the background ontology.

Since a database view is equivalent to a query, we may apply it to the ground instances of the background ontology to return a new table (denoted "*the instances of the ontology view*"); assuming the mapping is good, these new entity pairs will be good training instances for the target relation.

## 2.3 Filtering the Set of Candidates

Large ontologies may contain millions entities belonging to thousands of types and participating in hundreds of relations. Even if we bound the number of joins allowed in view definitions, there are still too many potential mappings for VELVET to consider them all. Therefore, some filtering is required to narrow down the search space.

An obvious criterion is that the candidate joined relation must contain at least one ground instance present in the corresponding target relation. For the moment, assume we have already solved the entity mapping problem (*i.e.*, we know that entity $\hat{e}$ in the target ontology corresponds to $e$ in the background ontology). We can think of the background ontology as a graph, where entities are nodes and binary relations are edges. If background entity pair $(e_1, e_2)$ maps $(\hat{e}_1, \hat{e}_2)$ of the target relation, we can return all paths between $e_1$ and $e_2$.[3] Suppose joined relation $r = r_1 \bowtie r_2 \bowtie \ldots \bowtie r_k$ is one path, then $r$ is a potential mapping to $\hat{r}$.

However, the entity mapping problem itself is hard. For example, suppose that (*Kobe Bryant, Phil Jackson*) is an instance of the target relation `isCoachedBy`, and let Freebase be the background ontology. There are more than 10 people named *Phil Jackson* in Freebase, including a football player, an author and a film crewmember.

Human beings can easily pick up the basketball coach among these 10 people because the environment (*e.g.* Kobe Bryant, coach)

[3]These paths may be easily generated using breadth first search.

disambiguates him. We can encode this intuition into our automated mapping algorithm by solving the entity mapping problem jointly with relation mapping.

For training instance $(\hat{e}_1, \hat{e}_2)$ of the target relation $\hat{r}(\hat{t}_1, \hat{t}_2)$, we create

- *Entity Mapping Candidates:* we return two sets of entities $E_1 = \{\ldots, e_{1,i}, \ldots\}$ and $E_2 = \{\ldots, e_{2,j}, \ldots\}$ found in background ontology by using IR search techniques on the names of $\hat{e}_1$ and $\hat{e}_2$.[4].
- *Type Mapping Candidates:* the background types of elements of $E_1$ and $E_2$ are type mapping candidates for $\hat{t}_1$ and $\hat{t}_2$, respectively.
- *Relation Mapping Candidates:* $r$ is the candidate for $\hat{r}$ if $(e_1^*, e_2^*) \in r$ where $e_1^* \in E_1$ and $e_2^* \in E_2$.

## 2.4 Constructing Mappings Jointly

In order to define a probability distribution over the space of possible joint mappings, we use Markov logic, a formalism which combines the expressiveness of first-order logic with the semantics of Markov networks [29]. Our model generates a set of rules with *predicates* and their negation. Each rule has some weight. VELVET encodes our interested mapping candidates into following predicates:

$$\hat{r} \cong r, \hat{e} \cong e, \hat{t} \cong t$$

$\hat{r} \cong r$ is true if the joined relation $r$ maps to the target relation $\hat{r}$. The same definition applies for $\hat{e} \cong e$ and $\hat{t} \cong t$.

An assignment to all predicates $\hat{r} \cong r$, $\hat{t} \cong t$ and $\hat{e} \cong e$ represents a possible ontology mapping result. By finding the best assignment that maximizes the weighted sum of all satisfied rules, we can create views for $\hat{r}(\hat{t}_1, \hat{t}_2)$ (*i.e.*, the target relation and its type constraints) as the union query:

$$\cup_{r,t_1,t_2} \pi_{t_1.\text{name},t_2.\text{name}} \quad r \tag{1}$$

where $\hat{t}_1 \cong t_1$, $\hat{t}_2 \cong t_2$ and $\hat{r} \cong r$ are true in the assignment.

Formally, let variables $\mathbf{X}$ represent truth assignments to all grounded predicates. We then model the joint probability distribution as

$$P(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(\mathbf{x})\right) \tag{2}$$

where $i$ ranges over our set of rules, $w_i$ denotes the weight of rule $i$, and $n_i(\mathbf{x})$ the number of true groundings of $i$ under assignment $\mathbf{x}$, $Z$ is the partition function $Z = \sum_{\mathbf{x}} \left(\sum_i w_i n_i(\mathbf{x})\right)$. Our goal is to find the MAP solution $\arg\max_{\mathbf{x}} P(\mathbf{X} = \mathbf{x})$.

We introduce our Markov logic rules below, and in Section 3 we present an inference algorithm to compute $\arg\max_{\mathbf{x}} P(\mathbf{X} = \mathbf{x})$.

### 2.4.1 Rules in Markov Logic

**Name similarity:** Related relations in target and background ontologies may have similar names. We compare $\hat{r}$ and $r$ by first tokenizing the relation names (splitting on camel-case transitions and punctuation), and then checking if they have some words $w$ in common. Formally, we encode this as a rule

$$\begin{aligned} &hasWord(\hat{r}, w) \wedge hasWord(r, w) \\ &\wedge candidate(\hat{r}, r) \Rightarrow \hat{r} \cong r \end{aligned} \tag{3}$$

[4]For Freebase, one can find these entities with its search API http://api.freebase.com/api/service/search?query=string

$hasWord(\hat{r}, w)$, $hasWord(r, w)$ are variables being true when word $w$ appears in the surface string of $\hat{r}$ or $r$ respectively. For example, word $w$ ="coach" appears in both $\hat{r}$ `:isCoachedBy` and $r$ : *BasketballTeamHeadCoach*, it suggests the later might be a good mapping.

For simplicity, we will from now on assume that the predicate $candidate(\hat{r}, r)$ is implicitly added as a pre-condition to every rule.

The name similarity rule can be applied to type and entity in the same way. Formally, they are

$$hasWord(\hat{e}, w) \wedge hasWord(e, w) \Rightarrow \hat{e} \cong e \tag{4}$$

$$hasWord(\hat{t}, w) \wedge hasWord(t, w) \Rightarrow \hat{t} \cong t \tag{5}$$

**Relation instance constraints:** The goal of VELVET is to create queries in the target ontology that contains entity pairs as good training instances to train the target relation's extractor. So it is nature to expect the training instances of the target relation $\hat{r}$ are also the instances of the background joined relation $r$.

We write the above intuition as

$$\begin{aligned} &(\hat{e}_1, \hat{e}_2) \in \hat{r} \wedge (e_1, e_2) \in r \wedge \hat{e}_1 \cong e_1 \wedge \hat{e}_2 \cong e_2 \\ &\Rightarrow \hat{r} \cong r \end{aligned} \tag{6}$$

where $(\hat{e}_1, \hat{e}_2) \in \hat{r}$ is true when $(\hat{e}_1, \hat{e}_2)$ is an instances of the target relation $\hat{r}$. Same notation is used in $(e_1, e_2) \in r$. By adding $\hat{e}_1 \cong e_1$ and $\hat{e}_2 \cong e_2$, we ensure $\hat{r}$ and $r$ are dealing with the same entity pair.

Sometimes the above intuitive rule suffers from a problems. Suppose $(\hat{e}_1, \hat{e}_2) \in \hat{r}$ and $(e_1, e_2) \in r$ is true, the conjunctive normal form of the Equation 6 is

$$\hat{e}_1 \ncong e_1 \vee \hat{e}_2 \ncong e_2 \vee \hat{r} \cong r \tag{7}$$

which means when $\hat{r} \cong r$ is false, the system tend to set at least one of $\hat{e}_1 \cong e_1$ and $\hat{e}_2 \cong e_2$ to be false. This is problematic. For example, when $\hat{r}$ is *stateHasCapital* and $r$ is *locationContains*; $\hat{e}_1$ and $e_1$ are *Massachusetts* in the target and background ontology; $\hat{e}_2$ and $e_2$ are *Boston*. (i.e. entity mapping is correct). Then, although $\hat{r} \cong r$ is false, it is not correct to suggest $\hat{e}_1 \ncong e_1$ or $\hat{e}_2 \ncong e_2$

Another weakness of the intuitive rule is that it does not handle overlapping relations. Suppose two joined relations $r_1, r_2$ contains one training instance of $\hat{r}$, the rule will lead to 2 formulas suggesting us to map $r$ into them both. Overlapped mapping results will not help relation extraction, and moreover, these formulas overemphasize one instance against other instances covered by fewer background relations.

With the above observations, we refine the intuitive rule into the following one:

$$\begin{aligned} &\hat{e}_1 \cong e_1 \wedge \hat{e}_2 \cong e_2 \wedge (\hat{e}_1, \hat{e}_2) \in \hat{r} \\ &\wedge (e_1, e_2) \in r_1 \wedge (e_1, e_2) \in r_2 \ldots \wedge (e_1, e_2) \in r_k \\ &\wedge (\hat{r} \cong r_1 \vee \hat{r} \cong r_2 \ldots \vee \hat{r} \cong r_k) \end{aligned} \tag{8}$$

We replace $\Rightarrow$ with $\wedge$ to avoid negative evidence on relation mapping force entity mapping. We use disjunction $\vee$ among $\hat{r} \cong r_i$ to avoid overlapping or overemphasizing the instance $(\hat{e}_1, \hat{e}_2)$.

One may wonder why Equation 8 is not symmetric between $\hat{r}$ and $r$. It is because usually the target ontology is small and its relations will not overlap, i.e. $(\hat{e}_1, \hat{e}_2)$ will not be an instance of $\hat{r}$ and $\hat{r}'$ that $\hat{r} \neq \hat{r}'$.

**Length of join:** While joining binary relations over the background ontology greatly extends the representational ability of the views, it may also add noise. The following rule encodes a prefer-

ence for views with fewer joins.

$$short(r) \Rightarrow \hat{r} \cong r \tag{9}$$

**Negative instances:** While many relations only contain positive examples, some ontologies embody the closed-world assumption or otherwise present negative examples and these can be powerful.

If such a negative target example is actully present in a background view $r$, then it is unlikely that $\hat{r} \cong r$ corresponds to the target.

$$\begin{aligned}&(\hat{e}_1, \hat{e}_2) \notin \hat{r} \wedge (e_1, e_2) \in r \wedge \hat{e}_1 \cong e_1 \wedge \hat{e}_2 \cong e_2 \\ \Rightarrow &\hat{r} \ncong r\end{aligned} \tag{10}$$

Unlike the Equation 8, we use $\Rightarrow$ because when $\hat{r} \cong r$, $(e_1, e_2) \in r$ but $(\hat{e}_1, \hat{e}_2) \notin \hat{r}$, it is very suspicious that the entity mapping is good, i.e. one of $\hat{e}_1 \cong e_1$ and $\hat{e}_2 \cong e_2$ could be false.

**Rank in search:** If the background ontology provides an entity search engine, the higher ranked returned entities are more likely to be good mappings. The rule can be written as

$$topSearch(\hat{e}, e) \Rightarrow \hat{e} \cong e \tag{11}$$

$topSearch(\hat{e}, e)$ means $e$ is the top returned entities by querying $\hat{e}$ on the search engine.

**Type constraints:** Suppose we already know the type of *Phil Jackson* (e.g. `coach`) maps to *basketballCoach* in the background ontology, we have more confidence to map *Phil Jackson* to an *basketballCoach* entity, rather than an *author* entity, even if two entities have the same name. Formally, the rule could be:

$$\hat{e} \in \hat{t} \wedge e \in t \wedge \hat{e} \cong e \wedge \hat{t} \cong t \tag{12}$$

We use $\hat{e} \in \hat{t}$ to denote the type of $\hat{e}$ is $\hat{t}$, and $e \in t$ to denote the type of $e$ is $t$. The above rule says we gain the weight when entity mapping $\hat{e} \cong e$ and type mapping $\hat{t} \cong t$ is consistent.

When the type of $e$ and $\hat{e}$ are unique in the ontologies, Equation 12 is good enough. But suppose the background entity $e$ has several type signatures, we should refine the rule into

$$\begin{aligned}&\hat{e} \cong e \wedge \hat{e} \in \hat{t} \wedge e \in t_1 \wedge e \in t_2 \ldots \wedge e \in t_k \\ &\wedge \left(\hat{t} \cong t_1 \vee \hat{t} \cong t_2 \ldots \vee \hat{t} \cong t_k\right)\end{aligned} \tag{13}$$

**Mutual exclusion:** For entity mapping, one can assume an entity in the target ontology maps to only one entity in the background ontology. We encode this as

$$\hat{e} \cong e \wedge e \neq e' \Rightarrow \hat{e} \ncong e' \tag{14}$$

For type mapping and relation mapping, there is no explicit mutual exclusion because VELVET may map these into a union of background types and relations. But one must notice there are already implicit mutual exclusions in Equation 8 and Equation 13. They help VELVET to avoid heavily overlapped mappings.

**Ockham Razor:** In practice, we prefer the assignment making the fewest number of predictions. We try to avoid predictions being true only with very weak evidence. For this purpose, VELVET adds rules with one negation predicate $\hat{r} \ncong r$ for relation candidate, $\hat{e} \ncong e$, $\hat{t} \ncong t$ for entity and type candidate.

# 3. MAP INFERENCE

We note that computing $\arg\max_{\mathbf{x}} P(\mathbf{X} = \mathbf{x})$ is challenging:

1. There exist thousands of grounded predicates, making exact inference intractable.
2. The dependencies represented by our rules break the joint distribution into islands of high-probability states with no paths between them.

For tractability we turn to approximate inference, and in particular to Gibbs sampling. In its basic version, Gibbs sampling generates a value for each variable in turn, conditioned on the other variables. In general, the samples then approximate the joint distribution.

In our scenario, however, the basic approach fails due to strong dependencies between variables. For example, $\hat{e} \cong e \wedge \hat{e} \ncong e'$ and $\hat{e} \cong e' \wedge \hat{e} \ncong e$ may both be possible states, but due to Rule 14 (mutual exclusion) Gibbs sampling cannot reach one state from the other.

We thus apply Blocked Gibbs sampling, where we directly sample from the true joint distribution of a subset of interdependent variables. For each entity $\hat{e}$ in the target ontology, we create a block that contains predicates $\hat{e} \cong e'$ for all entity mapping candidates $e'$ of $\hat{e}$. We create similar blocks for each type $\hat{t}$ and each relation $\hat{r}$.

Since we are only interested in determining the MAP assignment to $\mathbf{X}$, but not its probability, we ignore partition function $Z$ and score samples by the unnormalized objective, keeping track of the best sample found so far. For details see Algorithm 1.

**Algorithm 1**

**Input:** $\mathbf{X}$: variables for all predicates $\hat{r} \cong r$, $\hat{e} \cong e$ and $\hat{t} \cong t$,
$w_i$: weight for rule $i$,
$n_i(\mathbf{x})$: number of true groundings of rule $i$ under assignment $\mathbf{x}$,
$\mathbf{x}^{(0)}$: initial assignment,
$T$: number of samples
**Output:** $\mathbf{x}_{best} \approx \arg\max P(\mathbf{X} = \mathbf{x})$

```
x_best ← x^(0)
for j = 1 ... T do
  x^(j) ← x^(j−1)
  for all objects ô in target ontology do
    X_o ← {..., ô ≅ o_i, ...} where o_i all candidates for ô
    X_{−o} ← X \ X_o
    x_{−o} ← assignment to variables X_{−o} under x^(j)
    for all assignments x_o to X_o do
      p(x_o|x_{−o}) ← exp(Σ_i w_i n_i(x_o, x_{−o})) / Σ_{x'_o} exp(Σ_i w_i n_i(x'_o, x_{−o}))
    end for
    x*_o ← sample from distribution p(X_o|x_{−o});
    x^(j) ← (x*_o, x_{−o})
    if Σ_i w_i n_i(x_best) < Σ_i w_i n_i(x^(j)) then
      x_best ← x^(j);
    end if
  end for
end for
return x_best
```

The initial sample $\mathbf{x}^{(0)}$ is obtained greedily by considering only rules having one predicate, only considering atomic background relations (no views), and mapping a target object to only a single object in the background ontology.

# 4. RELATION EXTRACTION

After mapping the target ontology into the background knowledge, VELVET applies knowledge-based weak supervision [16] to heuristically match both the seed instances and the larger number of instances of the mapped relations, to corresponding text. For example, suppose that $r(e_1, e_2) =$ `Founded(Jobs,Apple)` is a ground tuple and $s =$"Steve Jobs founded Apple, Inc." is a sentence containing synonyms for both $e_1 =$ `Jobs` and $e_2 =$ `Apple`, then $s$ may be a natural language expression of the fact that $(e_1, e_2) \in r$ holds and could be a useful training example.

Unfortunately, this heuristic can often lead to noisy data and poor extraction performance. To fix this problem, Riedel et al. [30] cast weak supervision as a form of multi-instance learning, assuming only that *at least one* of the sentences containing $e_1$ and $e_2$ are expressing $(e_1, e_2) \in r$.

In our work, we use the publicly-available MultiR system [16] which generalizes Riedel *et al.*'s method with a faster model that also allows relations to overlap. For example, MultiR can consistently handle the fact that `Founded(Jobs, Apple)` and `CEO-of(Jobs, Apple)` are both true. MultiR uses a probabilistic, graphical model that combines a sentence-level extraction component with a simple, corpus-level component for aggregating the individual facts.

MultiR's extraction decisions are almost entirely driven by sentence-level reasoning. However, by defining random aggregate-level variables $Y$ for individual facts and tying them to the sentence-level variables $Z$ for extractions, a direct method for modeling weak supervision is provided. The model is trained, so that the aggregate variables $Y$ match the facts in the database, treating the sentence-level variables $Z$ as hidden variables that can take any value, as long as they produce the correct aggregate predictions.

During learning, MultiR uses a Perceptron-style additive parameter update scheme which has been modified to reason about hidden variables, similar in style to the approaches of [41, 19]. To support learning, MultiR performs a greedy approximation to a weighted, edge-cover problem for inference.

# 5. EXPERIMENTAL SETUP

In our experiments, we would like to show that ontological smoothing substantially improves the performance of an extractor. We would like to show that this is true across many target relations, each of which is only described by a small set of seed examples. Furthermore, we would like to separately investigate the performance of VELVET's crucial ontology mapping component.

## 5.1 Target Ontologies

We test VELVET on two different target ontologies, each of which makes extraction particularly challenging.

**Nell Ontology**: The Nell ontology [5], contains a total of 53 binary relations, each with a small number of positive seed examples. In addition, there also exist negative seed examples for many of the relations. Relations are unary or binary, and the arguments for binary relations are typed. In total, the seed instances cover 829 unique entities. There are 40 different entity types. The ontology contains some inconsistencies. For example, "Yankee", "Yankees" and "New York Yankees" appear as separate entities in different relations, although they point to the same real-world entity.

**IC Ontology**: The IC ontology is derived from the IC dataset which contains annotations of news articles that are relevant to the intelligence domain. For example, the dataset includes articles about terrorists and attacks. It is provided by the Linguistic Data Consortium [5]. In total, it covers annotations for 33 relations, but due to limitations of MultiR, we are currently only able to handle binary relations. We test VELVET for the 9 binary relations of the corpus: *attendedSchool*, *employs*, *hasBirthPlace*, *hasChild*, *hasCitizenship*, *hasSibling*, *hasSpouse*, *hasSubOrganization*, *isLedBy*. We collected instances for these relations from the annotated articles. We assumed gold coreference annotations, and replaced arguments that consisted of pronouns or nominals with their named entities. We allowed a maximum of 100 seed instances per relation. In total, the obtained IC ontology contains 388 positive seed examples, but no negative examples. This ontology too contains some inconsistencies, since the same named entities are sometimes referred to by different names. Furthermore, many of the annotated entities are less common ones.

[5]LDC2010E07, the Machine Reading P1 IC Training Data V3.1

## 5.2 Background Ontology

As our background ontology, we use Freebase [4] in its version [6] of May 2011. Freebase is ideally suited as a background ontology for various reasons:

- Freebase contains more than 3 million entities and tens of thousands of relations across a large number of domains. This makes it likely that there exists relevant information for virtually any target ontology.
- Freebase is organized as a database of triples of the form `<arg1, relation, arg2>`. This reduces the amount of ambiguity and enables easy processing.
- Freebase often contains synonyms for entities. Synonyms enable greater recall when heuristically matching instances to text.
- Freebase ranks candidate entities in its keyword search API. We use this ranking for identifying initial candidate mappings of entities in our target ontologies (our algorithm reduces the mapping error by a further 30%).

Despite its advantages, Freebase also poses important challenges: It makes heavy use of n:m helper relations in order to accurately represent its vast amount of knowledge. Even simple facts, such as a player's coach, are often not directly available, but can only be obtained by following a long chain of links, and merging the results of several relations. Furthermore, there exist redundant relations, a large number of irrelevant entities, while many important facts are missing.

## 5.3 Text Corpora

We evaluate VELVET on two text corpora, the New York Times and Wikipedia. The New York Times corpus [31] contains over 1.8 million news articles published between January 1987 and June 2007. The Wikipedia corpus covers more than 3.6 million encyclopedic articles in English language from May 2011.

## 5.4 Parameter Settings

The weights for VELVET's set of rules are currently set as follows: All weights are set to 1, except the weight for rule 10 and 14 which is set to 100. We use a high weight for that rule since we would not like to let it be violated. In future work, we would like to automatically learn weights for VELVET's set of rules.

To improve efficiency, we also limit the size of join computations [7] For example, the result set of *personBornInCountry* $\bowtie$ *countryHasCity* lists for each person all cities in her home country.

## 5.5 Overall Performance Metrics

To evaluate overall system performance, we run the full system which includes mapping the target ontology to the background ontology, generating training data using weak supervision, and learning an extractor.

Evaluating the quality of the learned extractor is challenging, however, since less than 1% of sentences in our large text corpora

[6]http://download.freebase.com/datadumps/latest/freebase-datadump-quadruples.tsv.bz2

[7]In particular, we remove candidate joins, if there exists a setting of the join attributes that yields more than $100 \times 100$ join tuples.

contain relations relevant to our target ontologies. We therefore compute two approximate metrics:

$\mathbf{M}_{\text{pre-rec}}$: For each relation in our target ontologies we manually create a view in our background ontology using the select, project, join, and union operators. We create that relation such that it most accurately matches the meaning of the corresponding relation in the target ontology. We then collect all instances of the newly created relation in the background ontology. Let this set be $B$. We next filter this set, keeping only those instances for which there exists a sentence in our text corpus $c$ that contains both arguments. Let us denote this filtered set by $\tilde{G}^c \subseteq B$. We use $\tilde{G}^c$ as an approximation to $G^c$, the set of facts contained in one of our text corpora $c$. To estimate precision and recall of our extractor, we simply compare its set of extractions $E^c$ on corpus $c$ to $\tilde{G}^c$. In practice, this approach provides a very conservative estimate of the quality of our extractor, since many facts in $\tilde{G}^c$ are not contained in our text corpora. We compute precision and recall curves by varying the confidence threshold of our extractor.

$\mathbf{M}_{\text{top-K}}$: To evaluate relation-specific performance of VELVET, we manually check the top-K extractions, for which our extractor is most confident. In our experiments, we set $K = 10$. To verify an extraction, we manually check all sentences which contain both arguments.

To ensure that we are testing the quality of the learned extractor independently of the entity pairs used during training, we further require that not only the sentences, but also none of the entity pairs at test time has been seen at training time.

As a baseline, we train the MultiR extractor using only the seed examples obtained from the target ontology, but without leveraging the mapping to our background ontology.

## 5.6 Ontology Mapping Metric

The ontology mapping component is VELVET's most important one, so we are also interested in evaluating its performance independently from relation extraction.

$\mathbf{M}_{\text{onto}}$: We investigate precision and recall for entity mapping, type mapping, and relation mapping by manually validating the individual decisions. Note that our algorithm does not always return a mapping element in the background ontology for an element in the target ontology. This often makes sense, since Freebase, although large, does not cover all entities, types, or relations.

As a baseline, we use a naive ontology mapper which does not perform joint inference, but merely uses Freebase's internal search API to map objects in the target ontology to objects in Freebase.

# 6. EXPERIMENTS

We first report on overall relation extraction performance, and then investigate relation-specific results. Finally, we report detailed results of VELVET's ontology mapping component.

## 6.1 Overall Performance

### *6.1.1 Overall Extraction Quality*

Figure 4 shows precision and recall curves for our two target ontologies, Nell and IC, as well as our two text corpora, the New York Times and Wikipedia. Note that the graphs have been generated using our conservative $\mathbf{M}_{\text{pre-rec}}$ metric, so actual precision and recall may be higher. VELVET reaches substantially higher precision and recall than our baseline, which uses the extraction algorithm but without leveraging the mapping to our background ontology. This is consistently true for all tested combinations of target ontologies and text corpora.

| Relation | Precision at top 10 | |
|---|---|---|
| | NYT | Wikipedia |
| *acquired* | 80% | 80% |
| *actorStarredInMovie* | 70% | 90% |
| *athletePlaysForTeam* | 100% | 100% |
| *bookWriter* | 30% | 60% |
| *cityLocatedInCountry* | 100% | 80% |
| *competesWith* | 90% | 100% |
| *hasOfficeInCountry* | 80% | 90% |
| *headquarteredIn* | 70% | 100% |
| *teamPlaysInLeague* | 60% | 100% |
| *teamWonTrophy* | 60% | 100% |

Table 1: Precision of the ten most confident predictions by VELVET for ten Nell relations. VELVET reaches good performance across relations.

The poor performance of our baseline may seem surprising, but can easily be explained: There are only few seed instances for each relation in the target ontology making it difficult to learn an extractor. Furthermore, not every seed instance in the target ontologies matches to a sentence in the text corpus, so that the available number of training sentences may be even smaller, for some relations less than 10. In contrast, ontological smoothing generates thousands of new training instances for our relation extractors.

Comparing the two target ontologies, we observe that our baseline is higher for the IC relations than the Nell relations. This is likely due to the fact that on average the IC ontology has more seed instances (about 43) than Nell relations (about 14) per relation. Comparing the two text corpora, we notice that VELVETś performance on Wikipedia is substantially higher than on the New York Times. One reason is that Wikipedia contains more factual knowledge, and more stylized and simpler language which simplifies the extraction task.

Perhaps surprising are the dips in precision in the low recall range, for example in the case of IC and Wikipedia. We manually checked the ten most confident extractions and found that they were all marked as incorrect by our approximate $\mathbf{M}_{\text{pre-rec}}$ metric, but actually all represented correct extractions of facts which were simply not present in Freebase. We therefore believe that if we were able to compare to the true facts contained in our text corpus this dip would be removed.

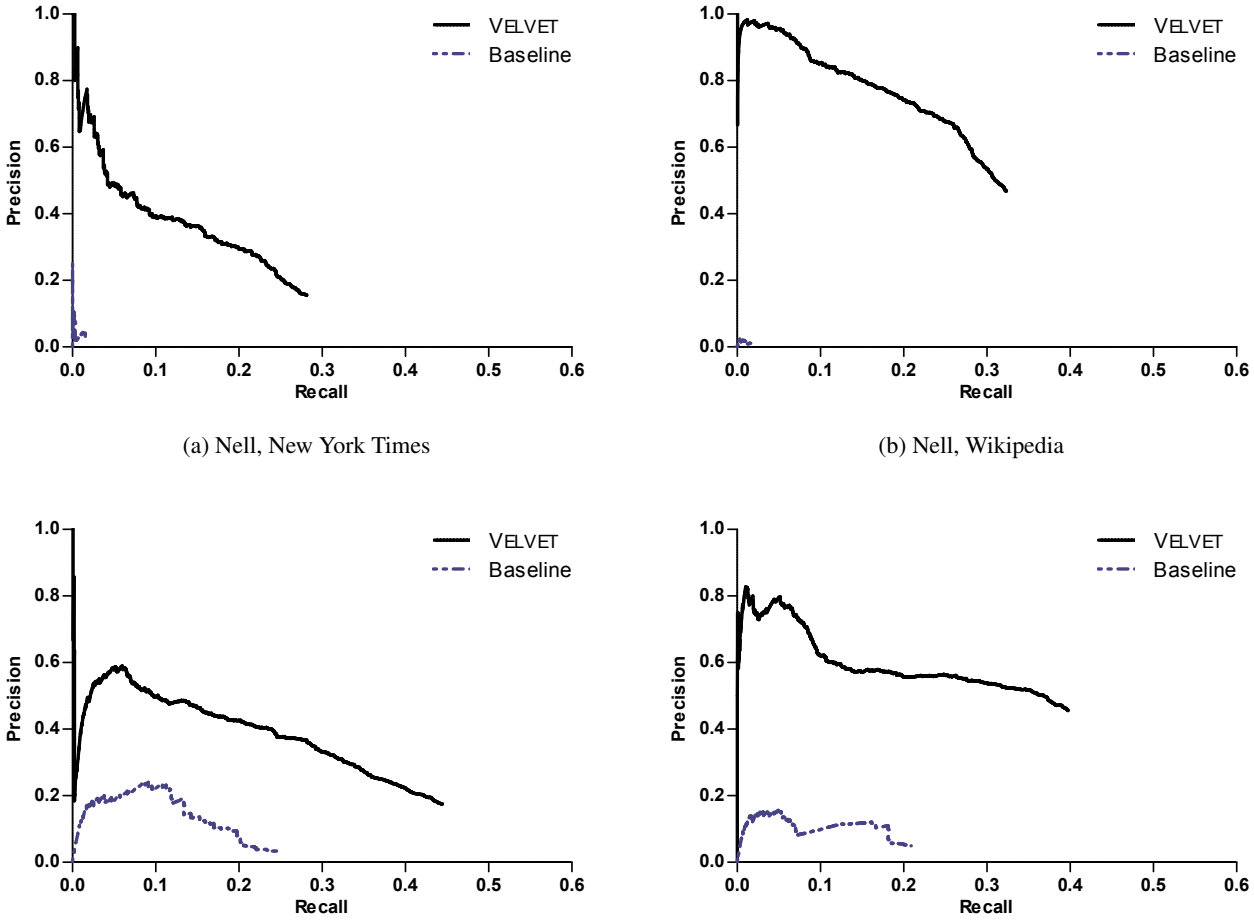
### *6.1.2 Relation-specific Extraction Quality*

To investigate relation-specific performance, we randomly picked ten relations of the Nell ontology and manually checked the top ten most confident extractions returned by our system ($\mathbf{M}_{\text{top-K}}$).

Table 1 presents the precision for each Nell relation and each text corpus. The majority of relations reach high precision at top-10: for the New York Times corpus the median is 80%, for Wikipedia it is 90%; the means are 74% and 91%, respectively. The results show that ontological smoothing makes it possible to learn accurate extractors from only a small number of seed examples, across many relations.

## 6.2 Ontology Mapping

Finally, we analyze the performance of our ontology mapping component in more detail. In particular, we are interested in knowing if our approach, which *jointly* maps entities, types, and relations, outperforms a baseline which relies on Freebase's internal search API and makes each mapping decision separately.




(a) Nell, New York Times

(b) Nell, Wikipedia

(c) IC, New York Times

(d) IC, Wikipedia

Figure 4: Approximate precision and recall of VELVET and a baseline that uses only the seed examples of the target ontology for knowledge-based weak supervision. VELVET consistently improves performance for two different target ontologies, Nell and IC, and two different text corpora, New York Times and Wikipedia.

We note that solving the mapping problem required finding a joint assignment to a considerable number of variables: For Nell, we computed truth values for 3055 entity mapping candidates, 252 type mapping candidates, and 729 relation mapping candidates. For IC, these are 1552, 130, and 256, respectively.

We then manually labeled 707 mappings of entities in Nell to entities in Freebase ($\mathbf{M}_{\text{onto}}$). VELVET reaches a precision of 92.79%, compared to 88.5% for our baseline. This corresponds to a reduction of 30% of mapping errors. Reducing mapping errors is important, since it leads to higher quality data for our weakly supervised extractors.

Table 2 shows the results of mapping five Nell relations to Freebase. VELVET is able to accurately recover relations composed by multiple select, project, join, and union operations.

For the IC target ontology, VELVET correctly maps 8 out of 9 relations. The results for the remaining relation *hasSubOrganization* are partially correct. The results show that our ontology mapping algorithm returns few incorrect mappings, thus ensuring the robustness of the overall system.

# 7. RELATED WORK

While Section 2.1 discussed previous approaches to ontology mapping, we now review work on using background knowledge to improve extractor learning and the exploiting ontologies for relation extraction.

## 7.1 Extraction with Background Knowledge

It has long been recognized that background knowledge can compensate for scare training data in machine learning. One such method is the use of constraints. Chang et al. [9] propose a technique for injecting prior knowledge into a semi-supervised learning algorithm as soft constraints. Constraints on two extraction tasks include the feature labels, *i.e.*, the relevance of words to particular labels, and also the number of times a label may appear.

More recent techniques incorporate background knowledge as expectations on the posterior distributions of an extractor model. Bellare & McCallum [3] obtain a 35% error reduction on a citation extraction task by adding expectations over how citation texts may align to a citation database and how a few features are highly indicative of a particular label. Chen *et al.* [10] propose a technique

| Target Relation | Mapped Relation |
|---|---|
| *acquired* | $\pi_{1.\text{name},2.\text{name}}(\text{businessOperation}^1 \bowtie \text{organizationChild} \bowtie \text{organizationRelationshipChild} \bowtie \text{businessOperation}^2)$<br>$\cup\ \pi_{3.\text{name},4.\text{name}}(\text{businessOperation}^3 \bowtie \text{organizationCompaniesAcquired} \bowtie \text{businessAcquisitionCompanyAcquired} \bowtie \text{businessOperation}^4)$ |
| *athleteCoach* | $\pi_{1.\text{name},2.\text{name}}(\text{baseballPlayer}^1 \bowtie \text{baseballCurrentTeam} \bowtie \text{baseballRosterPosTeam} \bowtie \text{baseballManager} \bowtie \text{coach}^2)$<br>$\cup\ \pi_{3.\text{name},4.\text{name}}(\text{footballPlayer}^3 \bowtie \text{footballTeam} \bowtie \text{footballRosterPosTeam} \bowtie \text{footballTeamHeadCoach} \bowtie \text{footballCoach}^4)$<br>$\cup\ \pi_{5.\text{name},6.\text{name}}(\text{basketballPlayer}^5 \bowtie \text{drafted} \bowtie \text{draftedTeam} \bowtie \text{basketballTeamCoach} \bowtie \text{coach}^6)$ |
| *bookWriter* | $\pi_{1.\text{name},2.\text{name}}(\text{book}^1 \bowtie \text{bookWrittenWorkAuthor} \bowtie \text{author}^2)$<br>$\cup\ \pi_{3.\text{name},4.\text{name}}(\text{book}^3 \bowtie \text{winAward} \bowtie \text{awardWinner} \bowtie \text{author}^4)$ |
| *headquarteredIn* | $\pi_{1.\text{name},2.\text{name}}(\text{businessOperation}^1 \bowtie \text{organizationHeadquarters} \bowtie \text{locationMailingAddressCitytown} \bowtie \text{citytown}^2)$ |
| *stadiumLocInCity* | $\pi_{1.\text{name},2.\text{name}}(\text{sportsFacility}^1 \bowtie \text{locationContainedby} \bowtie \text{cityTown}^2)$<br>$\cup\ \pi_{3.\text{name},4.\text{name}}(\text{olympicVenue}^3 \bowtie \text{locationContainedby} \bowtie \text{cityTown}^4)$ |

Table 2: VELVET Ontology mapping result on 5 Nell relations, with project, join, and union operators.

for relation discovery which uses expectations over the proportion of relation mentions matching certain syntactic patterns, the number of times a relation is instantiated, and the number of relation instances a single word can indicate.

While these approaches typically assume a small amounts of background knowledge supplied by a user, other approaches have tried to leverage existing resources as background knowledge. Stevenson & Greenwood [32] use WordNet to retrieve semantic relationships between lexical items in order to learn more general information extraction patterns. Cohen and Sarawagi [11] describe a technique for incorporating external dictionaries in discriminative sequence taggers. Other works by Wang *et al.* [35] and Hoffmann *et al.* [17] propose techniques to leverage the vast amount of structured lists on Web pages, in order to learn extractors with enhanced generalization ability. Both approaches apply a semi-supervised algorithm to learn extractor-specific lexicons.

VELVET is different from these approaches because it automatically generates the mapping to its background ontology, before applying semi-supervised techniques.

## 7.2 Using Ontologies for Extraction

A great deal of research has looked at automatically populating ontologies through extraction. A popular approach is by distant supervision, where existing objects in an ontology are heuristically aligned to a large text corpus, in order to create training data for an extractor. For example, Wu & Weld [39] and Hoffmann *et al.* [17] use Wikipedia's infobox ontology for distant supervision; Mintz *et al.* [24], Riedel *et al.* [30], and Hoffmann *et al.* [16] use Freebase.

Some work has proposed to leverage the hierarchical structure of an ontology for smoothing parameter estimates of a learned model. McCallum et al. [21] call this method *shrinkage* and demonstrate a 29% error reduction in a text classification task. Wu et al. [38] apply shrinkage to relation extraction for Wikipedia's infobox ontology, again showing large improvements. In this case, the hierarchical structure was not directly available, first necessitating ontological refinement [40].

Another direction applies reasoning over existing and new knowledge in order to disambiguate words and learn extraction patterns. Sofie [33] and Prospera [26] jointly perform pattern matching, word sense disambiguation and ontological reasoning in a unified model using weighted MaxSAT for inference. Similarly, Nell [6] couples the semi-supervised training of many extractors for different categories and relations through a variety of constraints.

Wimalasuriya & Dou [37] propose using multiple ontologies for extraction. Their system takes a mapping between the ontologies as input, and combines the output from extractors which have been learned separately learned on the ontologies.

VELVET differs from these approaches: The relations VELVET is able to extract are not limited to those in its background ontology. Instead, it automatically creates new relations by composing select, project, join, and union operations.

# 8. CONCLUSION

Relation extraction has the potential to enable improved search and question-answering applications by transforming information encoded in natural language on the Web into structured form. Unfortunately, the most successful techniques for relation extraction are based on supervised learning and require hundreds or thousands of training examples; these are expensive and time-consuming to produce. This paper presents ontological smoothing, a novel method for learning relational extractors, which requires only minimal supervision. Our approach is based on a new ontology-mapping algorithm, which uses probabilistic joint inference over schema- and instance-based features to search the space of views defined using SQL selection, projection, join and union operators. Experiments demonstrate the method's promise, improving both precision and recall. Our VELVET system learned significantly better extractors for 61 relations in two target ontologies, by exploiting Freebase as a background knowledge source.

# 9. REFERENCES


[1] Y. An, A. Borgida, and J. Mylopoulos. Discovering the semantics of relational tables through mappings. In *LNCS 4244 - Journal on Data Semantics VII*, pages 1–32, 2006.

[2] D. Aumueller, H.-H. Do, S. Massmann, and E. Rahm. Schema and ontology matching with coma++. In *IN SIGMOD CONFERENCE*, pages 906–908, 2005.

[3] K. Bellare and A. McCallum. Generalized expectation criteria for bootstrapping extractors using record-text alignment. In *EMNLP*, pages 131–140, 2009.

[4] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD Conference*, pages 1247–1250, 2008.

[5] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

[6] A. Carlson, J. Betteridge, R. C. Wang, E. R. H. Jr., and T. M. Mitchell. Coupled semi-supervised learning for information extraction. In *WSDM*, pages 101–110, 2010.

[7] M. J. Carman and C. A. Knoblock. Learning semantic descriptions of web information sources. In *International*

*Joint Conference on Artificial Intelligence*, pages 2695–2700, 2007.

[8] S. Castano and V. D. Antonellis. Artemis: Analysis and reconciliation tool environment for multiple information sources. In *SEBD'99*, pages 341–356, 1999.

[9] M.-W. Chang, L.-A. Ratinov, and D. Roth. Guiding semi-supervision with constraint-driven learning. In *ACL*, 2007.

[10] H. Chen, E. Benson, T. Naseem, and R. Barzilay. In-domain relation discovery with meta-constraints via posterior regularization. In *ACL*, 2011.

[11] W. W. Cohen and S. Sarawagi. Exploiting dictionaries in named entity extraction: combining semi-markov extraction processes and data integration methods. In *KDD*, pages 89–98, 2004.

[12] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. imap: Discovering complex semantic matches between database schemas. pages 383–394. ACM Press, 2004.

[13] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proceedings of the Eleventh International WWW Conference*, 2002.

[14] M. Ehrig and S. Staab. Qom Íc quick ontology mapping. In *In Proc. 3rd International Semantic Web Conference (ISWC04*, pages 683–697. Springer, 2004.

[15] J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.

[16] R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL*, 2011.

[17] R. Hoffmann, C. Zhang, and D. Weld. Learning 5000 relational extractors. In *ACL*, Uppsala, Sweden, July 2010.

[18] T. Lengauer, R. Schneider, P. Bork, D. L. Brutlag, J. I. Glasgow, H.-W. Mewes, and R. Zimmer, editors. *Constructing Biological Knowledge Bases by Extracting Information from Text Sources*. AAAI, 1999.

[19] P. Liang, A. Bouchard-Côté, D. Klein, and B. Taskar. An end-to-end discriminative approach to machine translation. In *International Conference on Computational Linguistics and Association for Computational Linguistics (COLING/ACL)*, 2006.

[20] J. Madhavan, P. Bernstein, and E. Rahm. Generic schema matching with cupid. In *In The VLDB Journal*, pages 49–58, 2001.

[21] A. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 359–367, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[22] R. J. Miller, L. M. Haas, and M. A. Hernández. Schema mapping as query discovery. In *VLDB*, pages 77–88, 2000.

[23] R. J. Miller, M. A. Hernandez, L. M. Haas, L. Yan, C. T. Howard, R. Fagin, H. Ronald, and L. Popa. The clio project: Managing heterogeneity, 2001.

[24] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *ACL-IJCNLP*, 2009.

[25] P. Mitra, N. F. Noy, and A. R. Jaiswal. Omen: A probabilistic ontology mapping tool. In *International Semantic Web Conference*, pages 537–547, 2005.

[26] N. Nakashole, M. Theobald, and G. Weikum. Scalable knowledge harvesting with high precision and high recall. In *WSDM*, pages 227–236, 2011.

[27] M. Niepert, C. Meilicke, and H. Stuckenschmidt. A probabilistic-logical framework for ontology matching. In *AAAI*, 2010.

[28] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB JOURNAL*, 10:2001, 2001.

[29] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 2006.

[30] S. Riedel, L. Yao, and A. McCallum. Modeling relations and their mentions without labeled text. In *Proceedings of the Sixteenth European Conference on Machine Learning (ECML-2010)*, pages 148–163, 2010.

[31] E. Sandhaus. *The New York Times annotated corpus*. Linguistic Data Consortium, 2008.

[32] M. Stevenson and M. A. Greenwood. A semantic approach to ie pattern induction. In *ACL*, 2005.

[33] F. M. Suchanek, M. Sozio, and G. Weikum. Sofie: a self-organizing framework for information extraction. In *WWW*, pages 631–640, 2009.

[34] J. D. Ullman. Information integration using logical views. In *ICDT*, pages 19–40, 1997.

[35] Y.-Y. Wang, R. Hoffmann, X. Li, and J. Szymanski. Semi-supervised learning of semantic classes for query understanding: from the web and for the web. In *CIKM*, pages 37–46, 2009.

[36] M. L. Wick, K. Rohanimanesh, K. Schultz, and A. McCallum. A unified approach for schema matching, coreference and canonicalization. In *KDD*, pages 722–730, 2008.

[37] D. C. Wimalasuriya and D. Dou. Using multiple ontologies in information extraction. In *CIKM*, pages 235–244, 2009.

[38] F. Wu, R. Hoffmann, and D. Weld. Information extraction from Wikipedia: Moving down the long tail. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-08)*, pages 731–739, New York, NY, USA, 2008. ACM.

[39] F. Wu and D. Weld. Autonomously semantifying Wikipedia. In *Proceedings of the ACM Sixteenth Conference on Information and Knowledge Management (CIKM-07)*, Lisbon, Porgugal, 2007.

[40] F. Wu and D. S. Weld. Automatically refining the Wikipedia infobox ontology. In *Proceedings of the 17th international conference on World Wide Web (WWW-08)*, pages 635–644, New York, NY, USA, 2008. ACM.

[41] L. Zettlemoyer and M. Collins. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-2007)*, 2007.