Ontological Smoothing for Relation Extraction with Minimal Supervision

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

Abstract

Relation extraction, the process of converting natural language text into structured knowledge, is increasingly important. Most successful techniques use supervised machine learning to generate extractors from sentences that have been manually labeled with the relations' arguments. Unfortunately, these methods require numerous training examples, which are expensive and time-consuming to produce.

This paper presents *ontological smoothing*, a semi-supervised technique that learns extractors for a set of minimally-labeled relations. Ontological smoothing has three phases. First, it generates a mapping between the target relations and a background knowledge-base. Second, it uses distant supervision to heuristically generate new training examples for the target relations. Finally, it learns an extractor from a combination of the original and newly-generated examples. Experiments on 65 relations across three target domains show that ontological smoothing can dramatically improve precision and recall, even rivaling fully supervised performance in many cases.

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* (RE). The most successful RE techniques use supervised machine learning to generate extractors from a training corpus comprised of sentences which have been manually labeled with the arguments of the target relations. 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.

To address this problem, researchers introduced the idea of *distant supervision*, a technique for automatically creating training data by heuristically matching the contents of a database relation to text (Craven and Kumlien). For example, if one has a table of athletes and their coaches that included the relation instance (`Jelani Jenkis`, `Urban Meyer`), then a system can automatically create a silver[1] training example for `isCoachedBy` from the following sentence: " 'Our captain, Jelani Jenkins, saved the day' said head coach, Urban Meyer."

[1] 'Silver' because these examples likely contain noise and aren't as valuable as 'gold standard' examples.

Figure 1: System overview

However, distant supervision only works when one has a large set of ground relation-instances (tuples) for the relation of interest. What can be done if a user wishes to quickly create an extractor, yet only has time to specify a handful of examples?

This paper presents VELVET, a novel technique called *ontological smoothing*, that addresses this problem, improving both precision and recall. VELVET learns extractors from a set of minimal labeled relations by exploiting a large background knowledge-base and unlabeled textual corpus. As shown in Figure 1, VELVET works in three phases: the first step uses the few examples to generate a mapping from the target relation to a database *view* over a background knowledge-base, such as Freebase. The second step queries the background knowledge-base to retrieve many more instances that are deemed similar to those of the the target relation; these are heuristically matched to the textual corpus to create myriad silver training examples. Finally, in the third step, VELVET learns an extractor.

It is challenging to find the best mapping from a target relation to a large background knowledge-base. Simply choosing the most similar background relation is insufficient. Instead, one should consider the large space of mappings formed by collections of database operations like join, union, project and select. For example, even though Freebase is extremely comprehensive, with considerable information about athletics, the relations have been broken into separate tables for individual sports, and the schemata have been normalized in a manner that eliminates a simple analogue to `isCoachedBy` (Figure 2). For this reason, and because of Freebase's massive size, it is challenging for an average user to construct good mappings manually. Often a

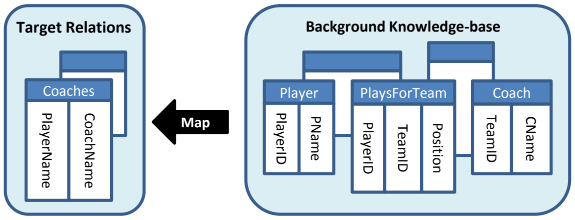


Figure 2: In order to map target relations to the background knowledge-base, one must consider a large space of possible database *views*. For example, the target isCoachedBy maps to the following expression over Freebase relations: $\pi_{\mathtt{PName},\mathtt{CName}}\mathtt{Players} \bowtie \mathtt{PlaysForTeam} \bowtie \mathtt{Coach}$. (In fact, the best mapping is a union of this expression with similar ones for other sports.)

user wishes to extract several interrelated relations. VELVET uses probabilistic joint inference over a set of Markov logic constraints to find the best global mapping.

In summary, VELVET makes the following contributions:

1. We introduce ontological smoothing, a novel approach for learning relational extractors given 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 database join, union, project and select operators.
3. We present experiments on 65 target relations across three ontologies, using Freebase as background knowledge, that demonstrate that ontological smoothing provides order-of-magnitude improvements over unsmoothed approaches and rivals fully supervised performance in many cases.

Constructing Ontological Mappings

The key intuition underlying ontological smoothing is that by finding a mapping from a user-specified target relation to a background knowledge-base, a system can automatically generate extra training data and improve the learned extractors. The key challenge is automatic construction of a good mapping from the target ontology to the background knowledge-base.

We assume that the target ontology is defined in terms of unary types T and binary relations R. We express the selectional preference (i.e. type constraint) of a binary relation by $\mathtt{R}(\mathtt{T}_1, \mathtt{T}_2)$. For example, isCoachedBy(athlete, coach) is a relation in the NELL ontology (Carlson et al. 2010a). We assume that each target relation comes with a set of labeled relation instances (tuples), denoted $\mathtt{R}(\mathtt{E}_1, \mathtt{E}_2)$. We also assume the presence of a large knowledge-base, $\mathcal{K}$, which is comprised of many types and relations and is populated with many instances (entities and ground relation instances); we denote these t, r, e, and $\mathtt{r}(\mathtt{e}_1, \mathtt{e}_2)$ respectively.

A *mapping* between a target relation, $\mathtt{R}(\mathtt{T}_1, \mathtt{T}_2)$, and $\mathcal{K}$, denoted $\phi(\mathtt{R}, \mathcal{K})$, is a SQL expression[2] over types and relations in $\mathcal{K}$'s schema; this expression defines a virtual relation, called a database *view*. Given a target ontology, some ground instances of its relations, and a background knowledge-base, the *ontology mapping problem* is the task of producing a mapping for each target, R, such that the instances of $\phi(\mathtt{R}, \mathcal{K})$ are semantically similar to those of R.

Ontology mapping is difficult because the space of possible views is huge. For example, Freebase contains more than 10,000 binary relations. Even if one restricts expressions to two joins with no unions or selections, there are more than 10^{12} possibilities. But selections are very important, as the following example illustrates. Suppose the target relation is stadiumInCity and consider following views:

$$\mathtt{SELECT}\ \mathtt{e}_1, \mathtt{e}_2\ \mathtt{FROM}\ \mathtt{containedBy} \quad (1)$$

$$\begin{aligned}&\mathtt{SELECT}\ \mathtt{e}_1, \mathtt{e}_2\ \mathtt{FROM}\ \mathtt{containedBy}, \mathtt{sportsFacility}, \mathtt{city}\\ &\quad \mathtt{WHERE}\ \mathtt{containedBy.e}_1 = \mathtt{sportsFacility.e}\\ &\quad \mathtt{AND}\ \mathtt{containedBy.e}_2 = \mathtt{city.e}\end{aligned} \quad (2)$$

The second view is a subset of the first and denotes a relation with very different semantics. In order for ontological smoothing to improve extractor performance, it's important to map as many ground instances as possible, but not too many! If VELVET mapped facts about cities in states and rivers in countries (as well as stadium locations), extractor precision would plummet.

To create good mappings, VELVET considers constraints between binary relations, unary types and entities — finding analogues for all three of these elements at the same time. We describe this process below, but one intuitive example is "If entity E in the target ontology corresponds to e in $\mathcal{K}$, then the type of E should correspond to the type of e." These constraints are described in Markov logic which combines the expressiveness of first order logic with a clear probabilistic semantics (Richardson and Domingos 2006).

At the highest level, VELVET uses a two-stage approach to find the best mappings. First, we restrict the set of views under consideration; this process, candidate generation, is described in the next subsection. Next, VELVET uses probabilistic joint inference to select the most likely global mapping from the candidates for each target relation, type and entity; our probability model and inference algorithm are described in the following subsections.

Generating Mapping Candidates

The first step in mapping construction is defining a set of *candidate mappings* for each of the target entities, types and binary relations; later these are ranked. Our model generates a set of Markov logic rules over special predicates and their negations: $\mathtt{Cnddt}(\mathtt{e}, \mathtt{E})$ means that the mapping between E and e is in consideration, and the probability of $\mathtt{Mp}(\mathtt{e}, \mathtt{E})$ signifies the quality of the mapping. We use a hard rule to ensure that two entities will only be mapped if they have similar names (Syn stands for synonym):

$$\mathtt{Syn}(\mathtt{e}, \mathtt{E}) \Rightarrow \mathtt{Cnddt}(\mathtt{e}, \mathtt{E}) \quad (3)$$

The next rule encodes the intuition that when two entities possibly match then their types might also match.

$$\mathtt{Cnddt}(\mathtt{e}, \mathtt{E}) \wedge \mathtt{Tp}(\mathtt{e}, \mathtt{t}) \wedge \mathtt{Tp}(\mathtt{E}, \mathtt{T}) \Rightarrow \mathtt{Cnddt}(\mathtt{t}, \mathtt{T}) \quad (4)$$

[2] We use a subset of SQL equivalent to relational algebra and sometimes use that notation for brevity; the symbols $\bowtie$, $\cup$, π and σ stand for database join, union, project and select operators.

Here, $\mathtt{Tp(e,t)}$ indicates $\mathtt{t}$ is the type of $\mathtt{e}$ in $\mathcal{K}$. The same notation applies for target terms: $\mathtt{Tp(E,T)}$.

We now turn to binary relations, such as $\mathtt{R(T_1,T_2)}$. VELVET only considers mapping $\mathtt{R}$ into views of the following form: $\cup\chi(\mathtt{t_1},\mathtt{t_2})$ where $\cup$ denotes union; χ is a join of up to 4 binary relations in $\mathcal{K}$; $\mathtt{t_i} = \phi(\mathtt{T_i})$ specify selection operations that only allow instances whose $\mathcal{K}$ entity arguments have types corresponding to the selectional preferences of the target, $\mathtt{T_i}$. Our next hard rule forces a candidate join path over $\mathcal{K}$ to contain at least one instance that is also present in the target relation.

$$\begin{aligned}&\mathtt{Inst(R,(E_1,E_2))} \wedge \mathtt{Inst}(\chi,(\mathtt{e_1},\mathtt{e_2}))\\ &\wedge\mathtt{Syn(e_1,E_1)} \wedge \mathtt{Syn(e_2,E_2)} \Rightarrow \mathtt{Cnddt}(\chi,\mathtt{R})\end{aligned} \tag{5}$$

The term, $\mathtt{Inst(R,(E_1,E_2))}$, means that the tuple, $\mathtt{R(E_1,E_2)}$, is a ground instance of target relation $\mathtt{R}$. $\mathtt{Inst}(\chi,(\mathtt{e_1},\mathtt{e_2}))$ means that $\mathtt{e_1}$ and $\mathtt{e_2}$ are elements in a row of χ, which was created by joining several relations from $\mathcal{K}$.

Our last hard rules specify that only candidates can be considered as mappings (we show the case for binary relations, but similar rules govern type and entity mappings):

$$\mathtt{Mp}(\chi,\mathtt{R}) \Rightarrow \mathtt{Cnddt}(\chi,\mathtt{R}) \tag{6}$$

Specifying the Likelihood of Mappings

We now describe our model for ascribing the probability of mappings. Here we use the full power of Markov logic. Unfortunately, our treatment must be brief. The probability of a truth assignment to the $\mathtt{Cnddt}$ and $\mathtt{Mp}$ predicates is given by $P(x) = (1/Z)exp(\sum_i w_i n_i(x))$, where Z is a normalization constant, w_i is the weight of the ith rule, and n_i is the number of satisfied groundings of the rule. See (Richardson and Domingos 2006) for details.

Consistency between Relations, Types and Entities: If many ground instances are shared between a target relation and its image under a mapping, then that suggests that the mapping is good. One might *think* that one could encode this as:

$$\begin{aligned}&\mathtt{Inst(R,(E_1,E_2))} \wedge \mathtt{Inst}(\chi,(\mathtt{e_1},\mathtt{e_2}))\\ &\wedge\mathtt{Mp(e_1,E_1)} \wedge \mathtt{Mp(e_2,E_2)} \Rightarrow \mathtt{Mp}(\chi,\mathtt{R})\end{aligned} \tag{7}$$

Unfortunately, this encoding causes problems. While this rule may look similar to Equation 5, this one affects the probability of both entity and relation mappings, since the probability of $\mathtt{Mp(e,E)}$ is also being inferred while synonyms (used in Equation 5) are taken as ground-truth inputs. The problem with Equation 7 is that it can cause VELVET to lower the probability of an (otherwise good) entity-entity mapping, whenever it dislikes a mapping between binary relations that involve those entities. Instead, we wish the inference to go one way: if many ground instances map, then the relations should be likely to map, but not vise versa. This is encoded as:

$$\begin{aligned}&\mathtt{Mp(e_1,E_1)} \wedge \mathtt{Mp(e_2,E_2)} \wedge \mathtt{Inst(R,(E_1,E_2))}\\ &\wedge \left(\vee_{\mathtt{k=1}}^{\mathtt{K}}\mathtt{Inst}(\chi_\mathtt{k},(\mathtt{e_1},\mathtt{e_2}))\right) \wedge \left(\vee_{\mathtt{k=1}}^{\mathtt{K}}\mathtt{Mp}(\chi_\mathtt{k},\mathtt{R})\right)\end{aligned} \tag{8}$$

Note that we've replaced $\Rightarrow$ with $\wedge$ to avoid negative "information flow." We use disjunction $\vee$ among $\mathtt{Mp}(\chi_\mathtt{k},\mathtt{R})$ to handle overlapped relations. Note Equation 8 is not symmetric between χ and $\mathtt{R}$; this is because the target ontology is usually small and its relations do not overlap. We specify a similar rule for types:

$$\mathtt{Mp(e,E)} \wedge \mathtt{Tp(E,T)} \wedge \left(\vee_{\mathtt{k=1}}^{\mathtt{K}}\mathtt{Tp(e,t_k)}\right) \wedge \left(\vee_{\mathtt{k=1}}^{\mathtt{K}}\mathtt{Mp(t_k,T)}\right)$$

Negative instance constraints: When specifying a target ontology, it is sometimes possible to declare a closed-world assumption, specify exclusion between types or otherwise present negative examples. Since these can greatly improve the quality of a mapping, we include the following hard rule:

$$\begin{aligned}&\mathtt{Inst}(\chi,(\mathtt{e_1},\mathtt{e_2})) \wedge \mathtt{NegInst(R,(E_1,E_2))}\\ &\wedge\mathtt{Mp(e_1,E_1)} \wedge \mathtt{Mp(e_2,E_2)} \Rightarrow \neg\mathtt{Mp}(\chi,\mathtt{R})\end{aligned} \tag{9}$$

Unlike the Equation 8, we use $\Rightarrow$ because when $\mathtt{Mp}(\chi,\mathtt{R})$, $\mathtt{Inst}(\chi,(\mathtt{e_1},\mathtt{e_2}))$ is true but $\mathtt{(E_1,E_2)}$ is a negative instance of $\mathtt{R}$, it is very unlikely that the entity mappings are correct.

Length of Join: While joining binary relations over the background ontology greatly extends the representational ability of the views, it may also add noise from arbitrary cross products. To combat this, we add a soft rule $\mathtt{short}(\chi) \Rightarrow \mathtt{Mp}(\chi,\mathtt{R})$, enforcing a preference for views with a small number of joins.

Unique Entities: We assume that the background knowledge base is of high quality, with little duplication among entities. This justifies the following hard rule: $\mathtt{Mp(e,E)} \Rightarrow \neg\mathtt{Mp(e',E)}$.

Regularization: According to Ockham's Razor, VELVET should avoid predictions with weak evidence. We add soft rules for type and relation mappings: $\neg\mathtt{Mp(t,T)}$ and $\neg\mathtt{Mp}(\chi,\mathtt{R})$. With respect to entity mappings, the unique entities rules achieve regularization.

Maximum a Posteriori Inference

Finding a solution to $\arg\max_\mathbf{x} P(\mathbf{x})$ is challenging. One issue is the scale of our problem: we would like to assign truth values to thousands of grounded predicates, but our problem, which is equivalent to the weighted Maximum Satisfiability problem, is NP-hard. Furthermore, the dependencies encoded in our rules break the joint distribution into islands of high-probability states with no paths between them — a challenge for local search algorithms.

One way of solving $\arg\max_\mathbf{x} P(\mathbf{x})$ is to cast it into an integer linear program (Motwani and Raghavan 1995). Although the integer linear program is intractable in our case, we can compute an approximation in polynomial time by relaxing the problem to a linear program and using randomized rounding, as proposed by (Yannakakis 1992). For solving the linear program we use MOSEK with the interior-point optimization method.

Relation Extraction

After mapping the target relations into the background knowledge-base $\mathcal{K}$, VELVET applies distant supervision (Craven and Kumlien 1999) to heuristically match both seed relation instances and relation instances of the mapped relations, to corresponding text.

For example, if $\mathtt{r}(\mathtt{e}_1, \mathtt{e}_2) = \mathtt{isCoachedBy}(\mathtt{Jenkins}, \mathtt{Meyer})$ is a relation instance and $\mathtt{s}$ is a sentence containing synonyms for both $\mathtt{e}_1 = \mathtt{Jenkins}$ and $\mathtt{e}_2 = \mathtt{Meyer}$, then $\mathtt{s}$ may be a natural language expression of the fact that $(\mathtt{e}_1, \mathtt{e}_2) \in \mathtt{r}$ 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. (Riedel, Yao, and McCallum 2010) cast distant supervision as a form of multi-instance learning, assuming only that *at least one* of the sentences containing $\mathtt{e}_1$ and $\mathtt{e}_2$ are expressing $(\mathtt{e}_1, \mathtt{e}_2) \in \mathtt{r}$.

In our work, we use the publicly-available MultiR system (Hoffmann et al. 2011) which generalizes Riedel *et al.*'s method with a faster model that also allows relations to overlap. MultiR uses a probabilistic, graphical model that combines a sentence-level extraction component with a simple, corpus-level component for aggregating the individual facts.

Training examples and their features are computed following (Mintz et al. 2009). On each sentence, we first run a statistical tagger to identify named entities and their types. Each pair of entity annotations is then considered as an extraction candidate, with features being conjunctions of the inferred entity types and paths of syntactic dependencies between the entity annotations.

For tagging named entities, we use the system by (Ling and Weld 2012). Since it outputs fine-grained entity types based on the Freebase type system, we can enforce consistency by considering only examples where the types of the tagger agree with those inferred in the mapping phase. We found that this step improves efficiency and leads to more accurate extractions. For computing syntactic dependencies we use Stanford Dependency Parser (Marneffe, Maccartney, and Manning 2006).

Experiments

In our experiments we examine (1) the impact of smoothing on the quality of relational extractors, (2) the quality of relation extraction using VELVET compared to supervised systems, and (3) the quality of ontological mappings inferred by VELVET.

Experimental Setup

In this paper, VELVET uses Freebase (Bollacker et al. 2008) as the background knowledge-base $\mathcal{K}$, which contains millions of entities and tens of thousands of relations across many domains. For the unlabeled corpus, we use the New York Times (Sandhaus 2008) which contains over 1.8 million news articles published between Jan. 1987 and Jun. 2007. For practicality, we make two simplifications. First, we set all weights for VELVET's soft rules to 1. In future work, we may further improve results using weight learning. Second, we limit the size of join computations. In particular, we remove candidate joins if there exists a setting of the join attributes that yields more than $10,000$ join tuples.

Relation Extraction with Smoothing

We compare VELVET to the following baseline conditions:

w/oS "without smoothed instances": Learns extractors from ground relation instances only; makes no use of background knowledge-base $\mathcal{K}$.

w/oC "without complex mappings": Maps each target relation to a single atomic relation in the background knowledge-base, that covers most ground relation instances. Type information is ignored. One-to-one mappings are also known as *alignments*.

w/oJ "without joint inference". Computes a complex mapping of target relations to the background knowledge-base involving $\bowtie$, π, and σ operators.[3] First, each target relation and each target type are assigned the background relations and types which cover most ground instances. Then, type constraints are enforced by taking appropriate joins.

We conduct experiments on relations of two target ontologies: NELL and IC. The **NELL ontology** (Carlson et al. 2010a)[4] contains 118 binary relations, but only 52 relations have a small number of positive ground instances. Many of these also have negative instances. The arguments for binary relations are typed. In total, the ground instances cover 40 different entity types and 829 unique entities. The **IC ontology** is derived from the IC dataset of the Linguistic Data Consortium[5]. The dataset contains annotations of news articles relevant to the intelligence domain. The IC ontology contains 9 binary relations, and we collected 388 positive ground instances from the annotated articles of the dataset.

We note that it is difficult to create a test set with enough gold annotations, since mentions of these 61 relations tend to be sparse. Thus we adopt the (semi-)automatic evaluation metric used in (Riedel, Yao, and McCallum 2010), which we call M_1. For each target relation, we estimate precision and recall by comparing two answer sets, Δ and Δ_V. Δ_V represents the set of predicted relation instances; Δ represents the set of relation instances in our background knowledge-base. In our work, we compute Δ by manually creating the best gold mapping from a target relation into the background knowledge-base using any combination of relational algebra operators, and then retrieving all instances. When aggregating over multiple relations, M_1 averages over instances.

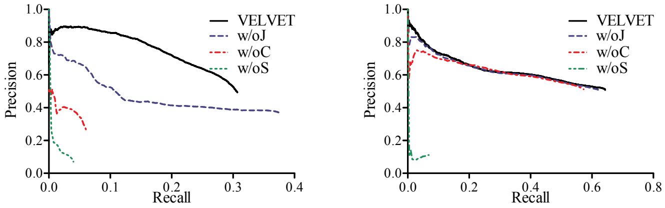
Figure 3 shows precision and recall curves. The poor performance of "w/oS" is due to the fact that there exist only few ground instances for each target relation, and often even fewer ground instances can be matched to sentences.

Smoothing, however, dramatically improves performance. We further observe that complex mappings are important: w/oC which only finds an alignment performs worse than w/oJ or VELVET. Upon inspection, we noticed that w/oC often maps to over-general relations. For example, background relation containedBy is mapped to target relation stadiumInCity. We therefore need type constraints, but not only type constraints: The fact that VELVET outperforms w/oJ shows that VELVET's abilities to do joint inference and support $\cup$ operators are also crucial.

[3] There is no obvious way to handle $\cup$ operators, without joint-inference or learning thresholds.

[4] $\mathtt{http://rtw.ml.cmu.edu/aaai10_online/relations.xls}$

[5] $\mathtt{LDC2010E07, theMachineReadingP1ICTrainingDataV3.1}$



(a) NELL (b) IC

Figure 3: Relation extraction with minimal supervision. VELVET outperforms baseline conditions on two target ontologies, Nell and IC.

Ontology	w/oS	w/oC	w/oJ	VELVET	Manual
`NELL`	7.2	18.1	25.1	**27.1**	31.6
`IC`	11.3	37.9	39.4	**40.9**	41.4

Table 1: Approximate F1 scores averaged by relations. VELVET outperforms baseline conditions on two target ontologies, NELL and IC. Condition "Manual" shows performance of an extractor trained on smoothed instances of the best manually constructed complex mapping from target relations to background knowledge-base.

Although M_1 allows (semi-)automatic evaluation on millions of sentences, it has two drawbacks: Since M_1 averages over instances, relations with many instances contribute more to the overall score than sparse relations. Furthermore, the metric only provides a conservative estimate of performance when the knowledge-base is incomplete. We therefore also evaluate VELVET using additional metrics.

Table 1 compares VELVET to our baseline conditions, averaged over relations rather than instances. The relative comparisons are consistent with our observations so far. Note, however, that averaging over relations tends to give lower numbers than averaging over instances, because the system can learn more accurately from relations with more instances.

Table 2 shows a breakdown of results per relation. Precision, recall, and F1 are estimated using the conservative metric M_1, but we also report top-K accuracy for $K = 10$. For each relation we took the top ten extractions for which our extractor was most confident and manually checked correctness. We obtained results in the high $70\% - 100\%$ range.

Comparing to Supervised Extraction

In this section, we want to show that VELVET can achieve performance comparable to state-of-the-art supervised approaches but with much less supervision. For this experiment, we choose a standard dataset for which there exist numerous annotations.

We use the **Conll04** relation extraction dataset[6] (Roth and Yih 2007). The sentences in this dataset were taken from the TREC corpus and were fully annotated with entities, types and relations. There are five relations and four entity types. We use the same experimental settings as previous

[6] `http://cogcomp.cs.illinois.edu/Data/ER/conll04.corp`

Relation	Rec	Pre	F1	Acc@top-10
`bookWriter`	31.8	43.5	36.7	100%
`headquarterIn`	19.1	60.1	28.9	90%
`isCoachedBy`	28.9	10.3	15.3	70%
`stadiumInCity`	51.9	77.6	62.6	100%
`attendSchool`	69.4	44.4	54.2	80%
`isLedBy`	33.8	49.7	40.2	100%

Table 2: Relation-specific Precision, Recall, F1 (estimated using M_1), and Accuracy at top-10 (checked manually) for 4 NELL and 2 IC relations.

Relation	VELVET			CP10	RY07
	Rec	Pre	F1	F1	F1
`Kills`	33.4	29.4	31.3	75.2	**79.0**
`LiveIn`	65.8	49.4	**69.2**	62.9	53.0
`LocatedIn`	64.0	65.4	**64.7**	58.3	51.3
`OrgBasedIn`	67.4	47.1	55.5	**64.7**	54.3
`WorkFor`	61.8	78.5	69.1	**70.7**	53.1

Table 3: VELVET achieves performance comparable to state-of-the-art supervised approaches RY07 and CP10, when there exists an appropriate mapping to its background ontology. While RY07 and CP10 need fully labeled sentences, VELVET learns with minimal supervision of just 10 ground instances per relation. Freebase does not offer an appropriate mapping for the `Kills` relation.

work (Kate and Mooney 2010; Roth and Yih 2007) to enable direct comparison. In this setting, there are 1437 sentences and about $18,000$ instances. However, unlike the supervised approaches, we only provide VELVET 10 ground instances per relation and no sentence-level annotations.

VELVET's ontology mapping component finds correct mappings for four relations, `LocatedIn`, `OrgBasedIn`, `WorkFor`, `LiveIn`, and correctly determines that Freebase does not offer an appropriate mapping for the `Kills` relation.

Table 3 compares VELVET's relation extraction performance to that of CP10 (Kate and Mooney 2010) and RY07 (Roth and Yih 2007). When VELVET finds correct mappings, it achieves comparable performance to the state-of-the-art supervised approach CP10 and RY07. However, VELVET achieves this result with only a small set of labeled ground instances, while CP10 and RY07 used more than one thousand labeled sentences. Of course, VELVET only works if the target relation has an analogue in the background KB.

Ontological Mapping Quality

Finally, we analyze the performance of our ontology mapping component in more detail. Note that solving the mapping problem requires 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 the IC domain, these are 1552, 130, and 256, respectively.

We investigate accuracy for entity, type and relation mappings by manually validating the individual decisions. Note that our algorithm does not always return a mapping element in the background knowledge-base $\mathcal{K}$ for an element in the target ontology. This often makes sense, since Freebase, al-

Target Relation	Mapped View
bookWriter	$\pi_{1.name,2.name}$/book/book^{1} $\bowtie$ /book/written_work/author $\bowtie$ /en/author^{2} $\cup$ $\pi_{3.name,4.name}$/film/film^{3} $\bowtie$ /film/film/story_by $\bowtie$ /en/author^{4}
headquarterIn	$\pi_{1.name,2.name}$/business/business_operation^{1} $\bowtie$ /organization/organization/headquarters $\bowtie$ /location/mailing_address/citytown $\bowtie$ /location/citytown^{2} $\cup\pi_{3.name,4.name}$/organization/organization^{3} $\bowtie$ /organization/organization/headquarters $\bowtie$ /location/mailing_address/citytown $\bowtie$ /location/citytown^{4}
isCoachedBy	$\pi_{1.name,2.name}$ /american_football/football_player^{1} $\bowtie$ /american_football/football_player/passing $\bowtie$ /american_football/player_passing_statistics/team $\bowtie$ /american_football/football_team/current_head_coach $\bowtie$ /en/head_coach^{2}$\cup\pi_{3.name,4.name}$ /en/basketball_player^{3} $\bowtie$ /basketball/basketball_player/team $\bowtie$ /basketball/basketball_roster_position/team $\bowtie$ /basketball/basketball_team/head_coach $\bowtie$ /basketball/basketball_coach^{4}
stadiumInCity	$\pi_{1.name,2.name}$/sports/sports_facility^{1} $\bowtie$ /location/location/containedby $\bowtie$ /location/citytown^{2}
attendSchool	$\pi_{1.name,2.name}$ /people/person^{1} $\bowtie$ /people/person/education $\bowtie$ /education/education/institution $\bowtie$ /education/university^{2}
isLedBy	$\pi_{1.name,2.name}$ /government/political_party^{1} $\bowtie$ /government/political_party/politicians_in_this_party $\bowtie$ /government/political_party_tenure/politician $\bowtie$ /government/politician^{2} $\cup\pi_{3.name,4.name}$ /location/country^{3} $\bowtie$ /government/governmental_jurisdiction/governing_officials $\bowtie$ /government/government_position_held/office_holder $\bowtie$ /government/politician^{4}

Table 4: VELVET ontological mapping result on 4 NELL and 2 IC relations, with join, union, project and select operators.

though large, does not cover all entities, types or relations. It turned out that VELVET achieves 87.9% accuracy on relation mapping, 90.9% on type mapping and 92.9% on entity mapping. As a baseline, we use a Freebase internal search API to map entities in the target ontology to entities in Freebase. This baseline gets 88.5% accuracy, which means joint inference in VELVET results in a reduction of 30% of entity mapping errors.

Table 4 shows the results of mapping six relations to Freebase. VELVET is able to accurately recover relations composed by multiple select, project, join, and union operations. The results show that our ontology mapping algorithm returns meaningful mappings, thus ensuring the robustness of the overall system.

Related Work

Learning extractors with minimal supervision Learning extractors from scarce training data has been an active area of research. Weakly supervised algorithms based on bootstrapping (Brin 1998; Agichtein and Gravano 2000) start with a small number of annotated seed data, and then iteratively generate extraction patterns (by matching seed data to text) and more seed data (by matching extraction patterns to text). While there are many successful examples of bootstrapping, avoiding semantic drift is challenging, especially in the case of unary relations. Large-scale systems therefore often use additional constraints (Carlson et al. 2010b), require extraction patterns as inputs, rely on manual validations between iterations, or focus on known typed entities as candidate arguments (Nakashole, Theobald, and Weikum 2011).

Other approaches target different kinds of background knowledge. (Chang, Ratinov, and Roth 2007; Smith and Eisner 2005; Bellare and McCallum 2009) allow learning with soft constraints, for example in the form of labeled features. (Stevenson and Greenwood 2005) use WordNet to learn more general extraction patterns, and (Cohen and Sarawagi 2004) use domain-specific dictionaries. (McCallum et al. 1998; Wu, Hoffmann, and Weld 2008) leverage the hierarchical structure of an ontology to smooth parameter estimates of a learned model.

VELVET differs from that work in that it uses a different kind of resource for background knowledge, Freebase, and more importantly in that it does not require any additional manual input besides the seed ontology. VELVET *automatically* recovers the mapping between seed ontology and background knowledge.

Mapping between ontologies Euzenat & Shvaiko (Euzenat and Shvaiko 2007) and Rahm & Bernstein (Rahm and Bernstein 2001) 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).

While the alignment problem has been studied extensively, far less work has looked has looked at finding complex mappings between ontologies. Artemis (Castano and Antonellis 1999) creates global views using hierarchical clustering of database schema elements. MapOnto (An, Borgida, and Mylopoulos 2006) produces mapping rules between two schemas expressed as Horn clauses. Miller *et al.*'s tool Clio (Miller, Haas, and Hernández 2000)(Miller et al. 2001) generates complex SQL queries as mappings, and ranks these by heuristics.

In our experiments, ontological smoothing only worked well when we allowed 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.

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, 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, that 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 65 relations in three target ontologies and rivals fully supervised performance in many cases.

Acknowledgements

We thank Xiao Ling for providing his fine-grained named entity tagging system, and we thank Oren Etzioni, Andrey

Kolobov, Luke Zettlemoyer, and the anonymous reviewers for helpful suggestions and comments. This material is based upon work supported by a WRF / TJ Cable Professorship, a gift from Google, ONR grant N00014-12-1-0211, and by the Air Force Research Laboratory (AFRL) under prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the Air Force Research Laboratory (AFRL).

References

Agichtein, E., and Gravano, L. 2000. Snowball: Extracting relations from large plain-text collections. In *In Proceedings of the 5th ACM International Conference on Digital Libraries*, 85–94.

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

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

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

Brin, S. 1998. Extracting patterns and relations from the world wide web. In *EDBT*, 172–183.

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

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

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

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

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

Craven, M., and Kumlien, J. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, 77–86.

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

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

Kate, R. J., and Mooney, R. J. 2010. Joint entity and relation extraction using card-pyramid parsing. In *COLING*.

Ling, X., and Weld, D. S. 2012. Fine-grained named entity tagging. In *AAAI*.

Marneffe, M.-C. D.; Maccartney, B.; and Manning, C. D. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC-2006)*.

McCallum, A.; Rosenfeld, R.; Mitchell, T. M.; and Ng, A. Y. 1998. Improving text classification by shrinkage in a hierarchy of classes. In *ICML*, 359–367.

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

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

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

Motwani, R., and Raghavan, P. 1995. *Randomized Algorithms*. Cambridge University Press.

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

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

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

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

Roth, D., and Yih, W. 2007. Global inference for entity and relation identification via a linear programming formulation. In Getoor, L., and Taskar, B., eds., *Introduction to Statistical Relational Learning*. MIT Press.

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

Smith, N. A., and Eisner, J. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *ACL*.

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

Wu, F.; Hoffmann, R.; and Weld, D. 2008. Information extraction from Wikipedia: Moving down the long tail. In *KDD*, 731–739.

Yannakakis, M. 1992. On the approximation of maximum satisfiability. In *Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, SODA '92, 1–9.