A Probabilistic Approach for Integrating Heterogeneous Knowledge Sources

Arnab Dutta, Christian Meilicke, Simone Paolo Ponzetto

Research Data and Web Science, University of Mannheim, Germany
{arnab,christian,simone}@informatik.uni-mannheim.de

Abstract. Open Information Extraction (OIE) systems like NELL and REVERB have achieved impressive results by harvesting massive amounts of machine-readable knowledge with minimal supervision. However, the knowledge bases they produce still lack a clean, explicit semantic data model. This, on the other hand, could be provided by full-fledged semantic networks like DBPEDIA or YAGO, which, in turn, could benefit from the additional coverage provided by Web-scale IE. In this paper, we bring these two strains of research together, and present a method to align terms from NELL with instances in DBPEDIA. Our approach is unsupervised in nature and relies on two key components. First, we automatically acquire probabilistic type information for NELL terms given a set of matching hypotheses. Second, we view the mapping task as the statistical inference problem of finding the most likely coherent mapping – i.e., the maximum a posteriori (MAP) mapping – based on the outcome of the first component used as soft constraint. These two steps are highly intertwined: accordingly, we propose an approach that iteratively refines type acquisition based on the output of the mapping generator, and vice versa. Experimental results on gold-standard data indicate that our approach outperforms a strong baseline, and is able to produce ever-improving mappings consistently across iterations.

Keywords: #eswc2014Dutta

1 Introduction

The last few years have witnessed much work in information extraction (IE). Although Wikipedia-based IE projects such as DBPEDIA [3] and YAGO [24] have been in development for several years, systems like NELL [6] and REVERB [12] have gained importance more lately. State-of-the art IE systems work on very large, i.e., Web-scale text corpora and are based on the general paradigm of *Open* Information Extraction (OIE) [2], which identifies IE systems that are not constrained by the boundaries of encyclopedic knowledge or a corresponding fixed schemata, unlike, for instance, those used by YAGO or DBPEDIA.

The data maintained by OIE systems is important for analyzing, reasoning about, and discovering novel facts on the web and has the potential to result in a new generation of web search engines [10]. However, while OIE systems have

very large coverage, they lack a full-fledged, clean ontological structure which, on the other hand, is essential in order to be able to exploit their output for Semantic Web applications. Often, the facts extracted by these systems are hard to decipher, and terms occurring in these very same facts can be highly ambiguous. For instance, let us consider a typical NELL extraction in the form of a *property(subject, object)* triple such as `agentcollaborateswithagent`(*knight, indiana*). While we might have an intuitive understanding of the property, it is difficult to determine the correct references of the terms in the triple. Actually, our example refers to 'Bob Knight', the head coach of the basketball team 'Indiana Hoosiers'. In contrast, an ontological resource, like DBPEDIA, uses a URI to uniquely identify each entity that appears within a triple. In our case, any DBPEDIA triple that talks about Bob Knight uses an unique URI to refer to that specific person. Thus, the meaning of a triple is precisely and uniquely specified.

In general, OIE systems trade-off large coverage for a weak, e.g., schema-less or schema-poor, semantic representation. In this work, we address this problem by bringing together information from a state-of-the-art OIE system and DBPEDIA. We achieve this by mapping the subject and object terms from NELL to DBPEDIA entities. Specifically, we propose a method to automatically determine the correct references of terms from OIE systems using probabilistic reasoning. We embed our probabilistic model that exploits the type hierarchy from DBPEDIA within a bootstrapping approach. As a result of this, we are able to provide via linking a clear semantic representation for both subject and object terms that occur within triples generated by a OIE system. Our hunch here is to provide a framework that makes it possible for different OIE projects to take advantage of the schema information provided by structured ontological resources like YAGO and DBPEDIA. This way the output of OIE is fully semantified within structured resources with an ontological model: by converse, the reference ontologies can benefit from the broader coverage of OIE projects.

2 Problem Statement

We present a methodology to map the output of OIE systems to an ontological resource like any of DBPEDIA, YAGO or FREEBASE. Key to our method is the synergistic integration of (i) information about the entity types the OIE terms can refer to and (ii) a method to find a global, optimal solution to the mapping problem across multiple extractions on the basis of statistical reasoning techniques. These two phases are highly intertwined, thus, we alternate between them by means of an iterative approach.

Given an OIE triple, there can be multiple plausible mappings to a set of highly related entities in the target ontology. For instance, the term *tom sawyer* occurring within the NELL triple `bookwriter`(*tom sawyer*, *twain*) can be mapped to a set of DBPEDIA entities: the fictional character Tom Sawyer (*Tom_Sawyer*), the actual book written by Mark Twain (*The_Adventures_of_Tom_Sawyer*), or the many screen adaptions of the book (e.g., *Tom_ Sawyer_(1973_film)*). While all these entities provide plausible meanings for the occurrence of *tom sawyer*,

knowing (or estimating) their types would allow us to further filter out meanings which are incompatible with the types of entities that occur as arguments of the property `bookwriter`. For instance, knowing that `bookwriter` relates books and authors would allow us to conclude that the correct mapping for *tom sawyer* in DBpedia is probably not a film or a fictional character, but rather an instance of a book. However, domain or range restriction in terms of DBpedia concepts are not defined for the extraction results of OIE systems. When including entity type information in the mapping problem, we are faced with two challenges, namely: (i) to estimate weights for the domain and range type of a Nell property term using the terminology of DBpedia; (ii) to effectively exploit this information in the actual mapping task.

With respect to the range of `bookwriter`, a good weight distribution would, for example, entail that the type `Writer` is more probable than `Politician` and `Politician` is more probable than `Location`. Note that its not sufficient to determine `Writer` as range of `bookwriter`, because many entities writing books are not explicitly typed as `Writer` but are of different types (e.g. `Athlete`s can also write books). Given a weight distribution for domain and range types of entities, we want to exploit this information in a way as to automatically identify the correct mappings. Using a statistical approach, we can start with some prior probabilities for each of the mapping candidates, and combine these priors with weighted type information such that we produce the best set of matches as output.

We show in the following that, for the resource mapping task at hand, acquiring type information and producing high-quality mappings are two highly intertwined problems. Our method starts with a set of mapping hypotheses between OIE terms and DBpedia instances. We combine these potential mappings with automatically learned entity type information (Section 3.1), and define a joint inference task within Markov Logic Network (Section 3.2). Furthermore, we propose a bootstrapping algorithm (Section 3.3) that generates better mapping hypotheses and refines the weight distribution for the learned types over a repeated number of iterations. In Section 4 we report about the experimental results. In Section 5 we provide an overview of related work and, finally, we conclude in Section 6 with scopes of possible extension.

3 Methodology

Our approach consists of three main phases. We first derive a distribution of weights over the possible domains and ranges of a given set of matching hypotheses for those triples that share the same property (Section 3.1). Next, we formalize the task of choosing a set of mappings from a set of candidates using Markov Logic Networks (Section 3.2). Finally, we use the previously explained components within a bootstrapping architecture in order to iteratively improve the final outcome (Section 3.3).

Our algorithm requires a set of matching hypotheses, also referred to as mappings, as input. In the rest of the paper, we use the notation $n{:}x \rightarrow d{:}y$ to refer

to such mappings, where *n:x* refers to a term that occurs within a NELL triple, and *d:y* to a DBPEDIA instance. x and y can refer to the s(ubject), p(redicate) and o(bject) of an arbitrary triple. We require mappings to be annotated with weights, which quantify the likelihood of an OIE term referring to a DBPEDIA instance. Intuitively, the higher the weight, the more likely it is that the mapping is true. An example of three different candidate mappings for the NELL term *n:hemingway* are the following ones.

$$+1.34 : \textit{n:hemingway} \rightarrow \textit{d:Ernest\_Hemingway}$$
$$-2.22 : \textit{n:hemingway} \rightarrow \textit{d:Hemingway,\_South\_Carolina}$$
$$-2.58 : \textit{n:hemingway} \rightarrow \textit{d:Hemingway\_(comics)}$$

The framework we propose in this paper is independent of the specific method of generating the initial mappings. However, the method should generate meaningful weights that can be interpreted in a probabilistic context. An example for such a method is presented in our previous work [9] where, we extract the *top-k* most frequent senses of OIE terms from the Wikipedia corpus.

3.1 Probabilistic Type Generation

Let us in the following consider NELL triples of the format *n:p*(*n:s*, *n:o*), namely consisting of a *n:s*(ubject) and *n:o*(bject) that share the same property *n:p*. For each triple we create the matching hypotheses both for *n:s* and *n:o* by applying an arbitrary matching method that generates mappings annotated with weights. We refer to this set of mappings as $\mathcal{H}$. We next select a subset of $\mathcal{H}$ that consists of only the *best* (*top-1*) candidate for each NELL term. The resulting set of matching hypotheses contains two mappings for each triple, namely *n:s* $\rightarrow$ *d:s'* and *n:o* $\rightarrow$ *d:o'*. In the rest of the paper, we denote this set of functional mapping hypotheses as $\mathcal{M}$, which is essentially a subset of $\mathcal{H}$.

In the next step, the types of *d:s'* and *d:o'* are used as markers for the domain and range restrictions of *n:p*. We distinguish between the direct and indirect type of an instance. Class C is a direct type of instance a, denoted by $C(a)$, if there exists no sub class D of C, denoted by $D \sqsubseteq C$, such that $D(a)$ exists. We count the direct type of each mapped DBPEDIA instance in $\mathcal{M}$. Finally, we obtain a distribution over the direct type counts for the possible concepts, both for the domain and range of *n:p*. Figure 1 depicts a snippet of the concept hierarchy for the range of the property `bookwriter`, where the nodes represent the concepts and the numbers (in non-bold) denote their direct type counts. The sum of the counts at a particular level do not add up to their parent node's count, since we are only counting the direct types of each instance.

Key to our method is the observation that an appropriate weight distribution helps us establish whether a certain candidate mapping is correct or not, according to the type of *d:s'* or *d:o'*. Considering the most frequent class as a hard domain/range restriction could potentially perform well, but this would fail to consider other instances writing books, e.g. philosophers, researchers or even athletes. On the other extreme, it seems rational to also count the indirect

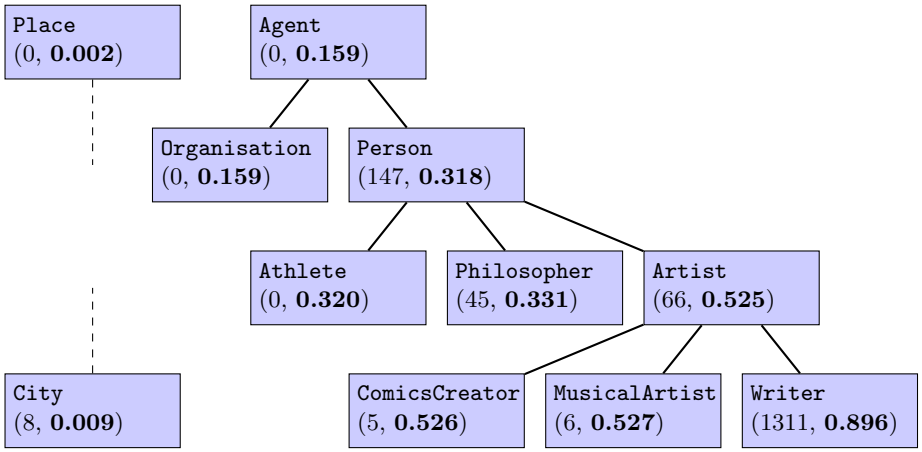

Fig. 1: Counting and weighing the range types of the `bookwriter` property. Each concept is accompanied by the counts of the direct types and the normalized S_d score for $\alpha = 0.5$ (shown in bold).

types or to propagate the count for a direct type recursively up to the parent nodes. However, this would result in a type restriction that takes only top-level concepts into account and completely disregards the finer differences expressed in the lower levels of the hierarchy. For example, a writer is more likely to write a book compared to an athlete. Accordingly, we opt for a hierarchical scaling of weights along the levels, such that the most likely class in the hierarchy is determined by the instance distribution of both its children and parent.

Hence, we propose a simple formulation to compute an appropriate score for each concept n. First, we introduce the *up-score*, S_u which is defined as

$$S_u(n) = S_o(n) + \alpha \sum_{c \in child(n)} S_u(c)$$

where $child(n)$ denotes the children of n, $S_o(n)$ refers to the direct type count and α is a constant, which works as a *propagation factor* with $\alpha \in [0, 1]$. The computation of this score starts from the leaf nodes, which are initialized with their direct count $S_o(n)$. S_u is defined recursively and, accordingly, the S_u score for n is computed based on the S_u score for the children of n. Furthermore, we also define a *down-score* S_d as

$$S_d(n) = \begin{cases} S_d(parent(n)) + (1-\alpha)S_u(n) & ; n \neq \textit{top-concept} \\ S_u(n) & ; n = \textit{top-concept} \end{cases}$$

where $parent(n)$ denotes the parent node of n. We refer to the concept hierarchy annotated with the S_d scores as the so-called α*-tree* in the rest of the paper.

We present in Figure 2 an example illustrating a simple hierarchy consisting of six concepts.The relevant scores for $\alpha = 0.5$ are shown adjacent to the nodes

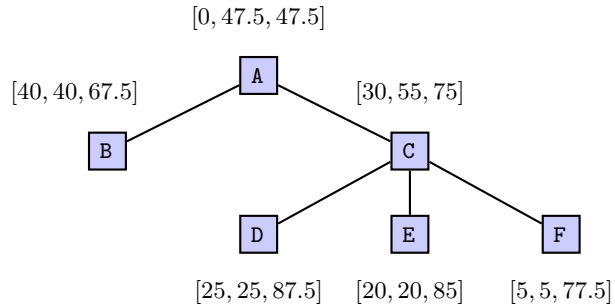

Fig. 2: Propagating direct counts in the alpha tree. Shown scores are $[S_o, S_u, S_d]$.

as $[S_o, S_u, S_d]$. This example illustrates that the sibling classes D, E and F, eventually, have the highest S_d scores, while the order among them, as defined by S_o, is still preserved in the order defined by S_d. As a final step, the down-scores are normalized by dividing them by the sum of the direct counts S_o for each node. With respect to Figure 2, the sum of S_o is $40+30+25+20+5 = 120$ and so the normalized S_d for node D, say, is estimated as a probability of $87.5/120 = 0.73$.

Obviously, the choice of the constant α is critical in achieving the desired result. Setting $\alpha = 0$, neutralizes the effect of child nodes on parent nodes. In this case we have $S_d(n) = S_o(n)$, which means that the type hierarchy is completely ignored. On the other extreme, setting $\alpha = 1$ propagates the scores to the full degree, but always creates the same scores for all concepts in the same branch. With respect to the example shown in Figure 1, we would learn that all concepts in the `Agent` branch have the same weight, while there are no differences between the concepts `Organisation` and `Writer`. In Section 4 we discuss the choice of the optimal α and report about experimental results related to different α values.

3.2 Modeling with Markov Logic Network

Markov Logic Networks (MLN) [23] are a framework for combining probability theory and first-order logic. Probabilities allow to quantify the uncertainty associated with complex processes and tasks, while first-order logic helps to capture the logical aspects of the problems. Formally, an MLN is a set of weighted first-order logic formulae. Under a set of constants, it instantiates into a ground Markov network where every node is a binary random variable and called a *ground atom*. In our task, we use three atoms to build the formulae of the MLN. We use `map` for mapping NELL terms to DBPEDIA instances, `isOfType` for specifying the types of the DBPEDIA instances, and `propAsst` for representing the NELL triples. Our set of constants is the set of NELL terms and the set of DBPEDIA instances that are the potential mapping candidates.

We can have several ($\approx 2^{\#groundings}$) network states for different boolean assignments of the ground atoms. Every such state is also called a *world*. In

those worlds different formulae hold true and if some do not, then the world is penalized according to the weights attached with the violating formulae. As a result, that world becomes less likely to hold (note that unweighted formulae can instead never be violated). According to [23], the probability of a world x is defined as $P(X = x) = \frac{1}{Z} exp(\sum_i w_i n_i(x))$ where w_i is the weight attached to the first-order logic formula F_i, $n_i(x)$ is the number of true groundings of F_i in x and Z is the normalizing factor (also called *partition function*) given as $\sum_{x \in \mathcal{X}} \prod_k \phi_k(x_{\{k\}})$, where, $\phi_k(x_k)$ is the feature function (usually binary) defined over the k^{th} clique in the ground network.

In our task, we employ both hard and soft formulae. The hard formula (one which cannot be violated) for our model is a restriction on the maximum number of mappings a particular NELL term can have. This is formally stated as

$$|\texttt{map}(n{:}t,\ d{:}t)| <= \quad 1$$

which denotes, for all possible instantiations of the `map` atom, that every NELL term *n:t* can have at most one mapping to a DBPEDIA instance *d:t*, i.e. we force the mapping to be functional. For each mapping candidate in $\mathcal{H}$, we add a weighted formula of the form

$$w : \texttt{map}(n{:}t,\ d{:}t)$$

where w is computed by applying the logit function[1] to the original probability awarded by the method that was used for generating $\mathcal{H}$. This adapts the weights to the underlying log-linear model of MLN.

To additionally take type information into account, we extend our model with two soft formulae for each possible combination of NELL property P and DBPEDIA type C. The first formula reflects a weighted domain restriction and the second formula reflects a weighted range restriction.

$$w_d(P, C) : \texttt{isOfType}(C,\ d{:}s) \wedge \texttt{propAsst}(P, n{:}s, n{:}o) \wedge \texttt{map}(n{:}s, d{:}s)$$
$$w_r(P, C) : \texttt{isOfType}(C,\ d{:}o) \wedge \texttt{propAsst}(P, n{:}s, n{:}o) \wedge \texttt{map}(n{:}o, d{:}o)$$

Note that P and C are replaced by constant values, while *n:s*, *n:o*, *d:s*, and *d:o* are quantified variables. $w_d(P, C)$ and $w_r(P, C)$ are the weights for type C with respect to property P computed with the α-tree as discussed in Section 3.1. If the type weight $w_d(P, C)$ is high, it makes the mapping *n:s* $\rightarrow$ *d:s* more likely in case that *n:s* appears in subject position of P and *d:s* is of type C.

Based on our model, we compute the MAP state, i.e., the most probable world which coincides in our scenario with the most probable mapping. As a result we select a subset $\mathcal{M}$ from the set of all mapping hypotheses $\mathcal{H}$. In particular, we want to select a better subset than just choosing the top-1 candidate for each NELL term. Note also that our model can easily be extended by adding more complex rules. The MAP inference is conducted with the RockIt system [20].

[1] The logit function is the inverse of the sigmoidal logistic function. For a probability P, the logit function is defined as $\log \frac{P}{1-P}$.

Algorithm 1 Bootstrapping algorithm

```
1: procedure BOOTSTRAP
2:     H ← set of mapping hypotheses
3:     M_0 ← top-1(H)
4:     i ← 0
5:     while M_i ≠ M_{i−1} do
6:         i ← i + 1
7:         T_i ← alphaTree(M_{i−1})
8:         M_i ← computeMAPState(H, T_i)
9:     end while
10:    return M_i                                  ▷ filtered output
11: end procedure
```

RockIt computes the most probable world by formulating the task as an Integer Linear Program. The solution of the resulting optimization is the MAP state of the Markov Logic Network.

3.3 Bootstrapping

So far, we used a subset of $\mathcal{H}$, namely the top-1 candidates, as input for computing the α-tree. Obviously, the quality of the chosen set of mappings directly impacts the quality of the resulting α-tree. At the same time, a better α-tree, represented as soft constraints in the MLN, can be expected to result in a better MAP state. Thus, we explore how to use the mapping that corresponds to the MAP state as input for constructing the α-tree and to use the resulting α-tree as input to recompute the MAP state.

We present our bootstrapping approach in Algorithm 1. The algorithm starts with the set of matching hypotheses $\mathcal{H}$. In the first iteration, we initialize $\mathcal{M}_0 \subset \mathcal{H}$ by selecting the top-1 candidates in $\mathcal{H}$. $\mathcal{M}_0$ is used as input to create the α-tree $\mathcal{T}_1$, which is then used together with $\mathcal{H}$ to generate the set of mappings for the next iteration, namely $\mathcal{M}_1$. Next, the algorithm checks if there is any difference between $\mathcal{M}_1$ and $\mathcal{M}_0$. If this is the case, the algorithm continues and repeats the same procedure, this time based on $\mathcal{M}_1$. With every iteration, each mapping going additionally into the hypothesis set, creates a refined α-tree, makes the added mapping more probable and so it stays in. Now, it cannot happen that it is added and then removed in subsequent iterations since it defies the reason for which it was added the first place. But, removal and then subsequent addition can happen. But once added, it stays. This continues as long as there are differences between $\mathcal{M}_i$ and $\mathcal{M}_{i-1}$ and terminates eventually. Finally, the last mapping set $\mathcal{M}_i$ that was generated in the i-th iteration is returned. Note that the functions *alphaTree* and *computeMAPState* refer to the modules described in Section 3.1 and Section 3.2 respectively.

4 Experiments

4.1 Metrics and Datasets

We apply the frequency based method as proposed in [9] to generate the set of input mappings $\mathcal{H}$ for our experiments. In the paper, we reported a micro-average precision of 82.78% and a recall of 81.31% for the subset defined by selecting the best candidate from $\mathcal{H}$. By choosing this subset as $\mathcal{M}_0$ in Algorithm 1, we are able to start with a set of input mappings that is already highly precise, thus ensuring high-quality information as seed for the bootstrapping method. We will use $\mathcal{M}_0$ as baseline and report about the improvements gained by each iteration of our algorithm.

The performance of our proposed approach is measured in terms of precision and recall. Let $\mathcal{M}$ refer to the set of mappings generated by our algorithm, and $\mathcal{G}$ refer to the set of mappings in the gold standard. Precision is defined as $|\mathcal{M} \cap \mathcal{G}|/|\mathcal{M}|$ and recall as $|\mathcal{M} \cap \mathcal{G}|/|\mathcal{G}|$. The F_1-measure is the equally weighted harmonic mean of both values. In particular, we compute these values for $\mathcal{M}_i$ after every i^{th} iteration (including the baseline $\mathcal{M}_0$).

We compare the results of our method with the gold standard $\mathcal{G}$, presented in [9].[2] This dataset has been created by randomly choosing twelve NELL properties. For each of these properties 100 triples have been sampled from the NELL dataset resulting in a rich set of 1200 triples for which subject and object mappings to DBPEDIA have been specified by human annotators.

4.2 Learning α

Our method relies on the choice of a proper value for the parameter α. In a first set of experiments, we analyze how to learn an appropriate α score. With respect to these experiments we report about results of our algorithm related to the final outcome of its last iteration. Experiments that focus on the impact of the different iterations are presented in Section 4.3.

Parameter Search: In the following we report about repeatedly performing a 2-fold cross validation on different samples of the whole dataset. For that purpose we restrict the possible values of α to a be multiples of 1/8 in the interval $[0, 1]$, which allows us to repeat the overall process over a large number of sampling steps (≈ 100000). At each sampling step, we first randomly pick half of the properties. This choice defines two datasets consisting of 6 properties (the chosen properties and the residual properties). We call one the training set D_{train} and the other the testing set, D_{test}. For every D_{train} we find the α giving the maximum averaged F_1 score over D_{train}. Then we apply our algorithm with that α on D_{test} and compute the resulting F_1. For 35% of the samples we learned α=0.5, for 30% we learned α=0.375, and for 18% we learned α=0.625. Approximately 85% of all samples yield an α in the interval $[0.375, 0.625]$, signifying that learning produces a stable outcome. Applying the learned α on D_{test} results in

[2] The dataset is publicly available at `https://madata.bib.uni-mannheim.de/65/`

Table 1: Effect of α on the overall performance compared to the baseline.

α	$prec$ (Δ_{prec})	rec (Δ_{rec})	F_1 (Δ_{F_1})
0.0	95.1 (+12.30)	76.1 (-5.20)	84.1 (+2.26)
0.125	94.8 (+12.04)	77.6 (-3.70)	84.9 (+3.08)
0.25	94.7 (+11.96)	78.5 (-2.80)	85.4 (+3.66)
0.375	94.4 (+11.58)	79.0 (-2.26)	85.6 (+3.84)
0.5	93.1 (**+10.34**)	79.9 (**-1.39**)	85.7 (**+3.94**)
0.625	92.3 (+9.48)	80.2 (-1.08)	85.6 (+3.76)
0.75	91.4 (+8.63)	80.4 (-0.96)	85.3 (+3.46)
0.875	90.3 (+7.53)	80.6 (-0.67)	85.0 (+3.15)
1.0	87.6 (+4.80)	81.0 (-0.35)	84.0 (+2.17)
Baseline	82.78	81.31	81.8

an increased average F_1 score of 85.74% (+3.94%) compared to the baseline with an average F_1 score of 81.8%. Thus, it is possible to learn α based on a small training set (600 triples) that results in a significant improvement of the mapping quality.

Parameter Effect: Finally we compute recall, precision and F_1 values on the entire dataset for all values of α in the $[0, 1]$ range with step sizes of 0.125. This helps us to better understand the impact of α on precision and recall. In Table 1 we report the absolute scores along with the differences (Δ) in scores over the most frequent baseline of [9] ($\mathcal{M}_0$ in Algorithm 1), and the output of the final iteration of the bootstrapped approach. Our results corroborate the findings from our cross-validation runs in that we achieve the best performance on the full dataset for $\alpha = 0.5$, which yields an improvement of +3.94% in terms of F_1 with respect to the baseline. Low values of α increase the precision by up to +12.3% ($\alpha = 0.0$), thus resulting in an overall precision of 95.1%, with a loss of 5.2% of recall as trade-off. While low values of α increase precision by aggressively eliminating many incorrect mappings, increasingly higher values lead to a drop in precision, indicating an ever increasing number of incorrect mappings being produced. This illustrates nicely that we can use α to adapt our approach to the needs of a certain application scenario, where precision or recall might be more important.

4.3 Algorithm Performance

In Table 2 we report the performance figures of our approach for each of the properties in the evaluation dataset. As baseline and inital starting point $\mathcal{M}_0$ we use the most frequent mapping presented in [9]. The following columns ($\mathcal{M}_i$, $i \neq 0$) report the F_1 scores of our proposed approach. For all experiments we set α=0.5. This choice is supported by the results of the previously-described cross-validation approach as well as by the theoretical considerations presented above. The results highlight that, thanks to our iterative approach, we are able to beat the baseline, and improve our results in average across iterations.

In our experiments, we found no improvements beyond the third iterations $\mathcal{M}_3$, thus indicating that our method quickly converges after few iterations.

Table 2: F_1 scores of the baseline ($\mathcal{M}_o$) and our bootstrapping approach, α=0.5

Nell Property	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$
actorstarredinmovie	81.3	87.7	94.5	-
agentcollaborateswithagent	83.7	76.4	82.0	-
animalistypeofanimal	85.9	86.0	86.7	-
athleteledsportsteam	87.0	92.6	93.2	-
bankbankincountry	79.6	86.4	83.4	82.8
citylocatedinstate	80.7	83.2	83.2	83.0
bookwriter	82.6	86.7	86.7	89.0
companyalsoknownas	64.1	64.6	67.1	-
personleadsorganization	76.7	78.1	77.2	77.6
teamplaysagainstteam	81.3	89.6	90.9	-
weaponmadeincountry	87.0	87.0	-	-
lakeinstate	91.4	94.4	94.7	-
Cumulative Gain (%)	-	2.62	3.89	3.94

Performance figures indicate the gradual saturation in the scores after each iteration. As expected, with each iteration the output gets more refined until a plateau in the results is reached, and no further improvement is gained with our method. In some cases our approach does not modify the original baseline (e.g. `weaponmadeincountry`). This is mostly attributed to missing type information in DBpedia. Note that results get slightly worse for some properties in some of the iterations. In the following section we analyze the behavior of our algorithm in details by looking at some concrete examples. These examples help to understand why some cases deviate from the general positive trend.

4.4 Analysis

First, we focus on the object of the Nell triple `actorstarredinmovie`(*al pacino, scarface*). The object term *scarface* has three possible mapping candidates, which are shown in the first column of Table 3 together with the case in which no candidate is chosen (identified by `None` in the table). In the table, we identify the candidate chosen in each iteration with a grey cell. `None` is chosen if the sum of all weights is less than 0, which means that all mapping candidates have a probability of less than 50%. For the column baseline, the only relevant weight corresponds to the most frequently-linked mapping candidate. No type-related weights are available, and accordingly the wrong entity *Scarface_(rapper)* is chosen. In the first iteration, the weights for the types are added to those of the mapping hypothesis. These type weights are obtained by applying the α-tree computation to the baseline. With respect to our example, this results in rejecting all of the candidates, because each has a probability of less than 50%. Specifically, we observe that the weight attached to the range type `Film` is not yet high enough to increase the overall score for the two movies up to a score greater than 0. The second iteration uses the type weights computed on the refined α-tree, which is created on the basis of the outcome from the previous iteration. The weights attached to `Film` have now increased significantly, and consequently one of the two movies is chosen. In this case, the algorithm chooses the right candidate. However, this example also reveals the limits of our approach, namely

Table 3: Weight refinements across iterations for the object of the triple `actorstarredinmovie`(*al pacino, scarface*) and for the subject of the triple `bookwriter`(*death of a salesman, arthur miller*). Grey cells refer to the mappings generated at each iteration.

Candidate	Type	Baseline	1^{st} Iteration	2^{nd} Iteration
Scarface_(rapper)	`MusicalArtist`	0.06	0.06 − 3.04=−2.98	0.06 − 13.81=−13.75
Scarface_(1983)	`Film`	−0.58	−0.58 + 0.36=−0.22	−0.58 + 3.37=2.79
Scarface_(1932)	`Film`	−2.22	−2.22 + 0.36=−1.86	−2.22 + 3.37=1.15
`None`	`[-]`	0.0	0.0 + 0.0=0.0	0.0 + 0.0=0.0
Salesman	`Play`	1.59	1.59 + 0.08=1.67	1.59 + 0.83=2.42
Salesman_(1951)	`Film`	−2.50	−2.50 − 0.05=−2.55	−2.50 + 0.57=−1.93

the fact that our method solely relies on type-level information. For this reason, the final choice between the movie from 1983 and the movie from 1932 is only based the fact that the movie from 1983 has a higher mapping weight, namely it is more often referred to by the surface form *Scarface*. That is, all things being equal (i.e., given the same type-level information), our approach will still choose the most popular entity, which might be a wrong choice.

The second example is the mapping of the subject term in `bookwriter`(*death of a salesman, arthur miller*). In this example, the candidate chosen by the baseline is also that chosen in each subsequent iterations. Contrary to the first example, the type of the chosen candidate is not the highest scoring type according to the α-tree, namely `Book`. While for the second iteration the weight for `Book` is +1.62, the weight for `Play` is +0.83, based on the fact that `Play` is a sibling of `Book`. Thus, its weight is not only supported by all matched plays, but also indirectly by all matched books. This example illustrates the benefits of the α-tree and the differences of our approach compared to an approach that simply uses the majority type as a hard restriction.

5 Related Work

Information Extraction: Key contributions in Information Extraction from the past years have concentrated on minimizing the amount of human supervision required in the knowledge harvesting process. To this end, much work has explored unsupervised bootstrapping for a variety of tasks, including the acquisition of binary relations [4], facts [11] and instances [21]. Open Information Extraction further focused on approaches that do not need any manually-labeled data [12] however, the output of these systems still needs to be disambiguated to entities and relations from a knowledge base. Recent work has extensively explored the usage of distant supervision for IE, namely by harvesting sentences containing concepts whose relation is known, and using them as training data for supervised extractors [27]. High-quality data can then be ensured by means of heuristics based on syntactic constraints [27] or encyclopedic content [14].
Matching Candidates: For over quite some time, researchers have made considerable efforts in solving the tasks of Entity Linking (EL) [15] and Word

Sense Disambiguation (WSD) [17]. Seminal work in EL includes contributions by Bunescu and Paşca [5] and Cucerzan [7], who focused on the usage of Wikipedia categories and global contexts, respectively. The Silk framework [26] discovers missing links between entities across linked data sources by employing similarity metrics between pairs of instances. Dredze et al. [8] achieved remarkable results using supervised approaches, in which they were able to link entities with missing knowledge base entries. Similarly for WSD, supervised systems have been shown to achieve the highest performance, although questions remain on whether these approaches perform well when applied to domain-specific data [1]. Besides, recent work indicates that knowledge-based methods can perform equally well when fed with high-quality and wide-coverage knowledge [22, 18]. In contrast to all these approaches, our method employs the most frequent sense of a term from Wikipedia. This consists of a simple, yet high-performing baseline that provides us with high-quality seeds for our bootstrapping approach.

Weighing Domain and Range: The task of learning domain and range weights closely matches with the ontology learning task. In this regard, association rule mining techniques have been employed to learn the axioms of an ontology by Völker er. al. [25]. Our way to generate weighted domain and range restrictions can be considered as a special case of ontology learning where we also assign a probabilistic rank to each of the concepts. However, our algorithm computes the weighted domain and range restrictions not as an end in itself, but as a means for improving the mapping quality between terms and instances.

Reasoning and Optimization Techniques: Niepert et.al. [19] focused in previous work on probabilistic alignment of ontologies: in our work, we focus instead on refining instance mappings by exploiting an ontological backbone. Recently, Galárraga et. al. [13] tackled the task of integrating knowledge bases (KB) by aligning instances across different KBs. Their approach involves the combination of multiple KBs into one, and learning logical rules using rule mining techniques. We have a minor overlap with their work in aligning instances across KBs. Our work of finding inconsistencies in a KB is more closely related to that of Jian et al. [16], who also use Markov Logic Networks to refine a knowledge base. Jian et al. also rely on data from NELL. However, in their work they manually assign weights to first order logic rules, whereas we use only a single parameter α, which can be learned in an automated way. Wang et. al [28] have employed (ProbKB) MLN for the task of automated KB creation through deduction and using parallel Gibbs sampling, which sets it different from our task of refinement.

6 Conclusions and Future Work

We presented a probabilistic approach to link terms from an OIE system to instances of a semantic resource in order to unambiguously determine the meaning of terms occurring within triples generated by an OIE system. In particular, we introduced a way to learn and assign weights to the possible domain and range types, defined in the terminology of the semantic resource, of a property from the OIE system. We formalized the task as a Markov Logic Network and solve

the resulting optimization problem as a MAP inference task. As a result, we are able to achieve highly refined mappings in terms of both precision and recall. Moreover, we have shown how to extend this approach in an iterative way to improve the results across iterations. Our method does not use any specific parameter settings apart from the propagation factor α, which is automatically learned from our data. Based on our iterative approach, we are able to increase the F_1-measure from 81.8% to 85.74%.

In the future, we plan to extend our method to exploit more than just the type hierarchy, where alternatives are typed with the same or a similarly weighted concept. In particular, we plan to jointly reason combinations of candidates for object and subject terms of a given triple by exploiting their attached properties. For instance, two instances in a `bookwriter` relation cannot have a negative difference between the book's publishing year and the writer's birth year. To this end, we will explore kernel density estimation techniques to learn the typical relation between the relevant data values. Note that this component can easily be integrated in the iterative framework, working as a booster for the type acquisition component and vice versa.

We will apply the overall framework on other OIE datasets, ReVerb [12] in particular. While NELL has a fixed number of cleaned properties, this is not the case for ReVerb. A property like `bookwriter` might be expressed by terms like `written by`, `has author`, or by an inverse term like `is author of`. The challenging task herein is to cluster corresponding property terms and apply the proposed mechanism. Finally, we plan to run more comprehensive evaluations with the aim to re-build a significant part of, e.g., DBPEDIA using the triple sets generated by different OIE systems. Ultimately, our vision is to extend knowledge bases like DBPEDIA with a rich set of highly precise novel RDF triples that fully exploit the potential of OIE systems. Our current results are promising in that they indicate that this could lead to a highly beneficial solution to synergistically exploit IE and OIE systems together.

Acknowledgments

We thank Mathias Niepert for his contributions and valuable feedback on this work. The authors gratefully acknowledge the support of a Google Faculty Research Award (see `http://dws.informatik.uni-mannheim.de/en/projects/current-projects/` for details).

References

1. E. Agirre, O. L. de Lacalle, and A. Soroa. Knowledge-based WSD on specific domains: performing better than generic supervised WSD. In *Proc. of IJCAI-09*, 2009.
2. M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the Web. In *Proc. of IJCAI-07*, 2007.

3. C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia – A crystallization point for the web of data. *Journal of Web Semantics*, 7(3), 2009.
4. S. Brin. Extracting patterns and relations from the World Wide Web. In *Proc. of WebDB Workshop at EDBT-98*, 1998.
5. R. Bunescu and M. Paşca. Using encyclopedic knowledge for named entity disambiguation. In *Proc. of EACL-06*, 2006.
6. A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *Proc. of AAAI*, 2010.
7. S. Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *Proc. of EMNLP-CoNLL-07*, 2007.
8. M. Dredze, P. McNamee, D. Rao, A. Gerber, and T. Finin. Entity disambiguation for knowledge base population. In *Proc. of COLING-10*, 2010.
9. A. Dutta, M. Niepert, C. Meilicke, and S. P. Ponzetto. Integrating open and closed information extraction : Challenges and first steps. In *Proc. of the ISWC-13 NLP and DBpedia workshop*, 2013.
10. O. Etzioni. Search needs a shake-up. *Nature*, 476(7358):25–26, 2011.
11. O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-scale information extraction in KnowItAll (Preliminary results). In *Proc. of WWW*, 2004.
12. A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proc. of EMNLP-11*, 2011.
13. L. A. Galárraga, N. Preda, and F. M. Suchanek. Mining rules to align knowledge bases. In *Proc. of AKBC-13*, 2013.
14. R. Hoffmann, C. Zhang, and D. S. Weld. Learning 5000 relational extractors. In *Proc. of ACL-10*, 2010.
15. H. Ji and R. Grishman. Knowledge base population: Successful approaches and challenges. In *Proc. of ACL-11*, 2011.
16. S. Jiang, D. Lowd, and D. Dou. Learning to refine an automatically extracted knowledge base using markov logic. In *Proc. of ICDM-12*, 2012.
17. R. Navigli. Word Sense Disambiguation: A survey. *ACM Computing Surveys*, 41(2):1–69, 2009.
18. R. Navigli and S. P. Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250, 2012.
19. M. Niepert, C. Meilicke, and H. Stuckenschmidt. A probabilistic-logical framework for ontology matching. In *Proc. of AAAI*, 2010.
20. J. Noessner, M. Niepert, and H. Stuckenschmidt. RockIt: Exploiting parallelism and symmetry for map inference in statistical relational models. In *Proc. of AAAI*, 2013.
21. M. Paşca and B. Van Durme. Weakly-supervised acquisition of open-domain classes and class attributes from Web documents and query logs. In *Proc. of ACL-08*, 2008.
22. S. P. Ponzetto and R. Navigli. Knowledge-rich Word Sense Disambiguation rivaling supervised systems. In *Proc. of ACL-10*, 2010.
23. M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2), 2006.
24. F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge. In *Proc. of WWW-07*. ACM Press, 2007.
25. J. Völker and M. Niepert. Statistical schema induction. In *Proc. of ESWC-11*, 2011.

26. J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Silk - A Link Discovery Framework for the Web of Data. In *Proc. of LDOW '09*, 2009.
27. F. Wu and D. Weld. Open information extraction using Wikipedia. In *Proc. of ACL-10*, 2010.
28. D. Z. W. Yang Chen. Web-scale knowledge inference using markov logic networks. *ICML workshop on Structured Learning: Inferring Graphs from Structured and Unstructured Inputs*, 2013.