Enriching Structured Knowledge with Open Information

Arnab Dutta, Christian Meilicke, Heiner Stuckenschmidt
Data and Web Science Research Group
Chair of Artificial Intelligence, University of Mannheim
Mannheim, 68159, Germany
{arnab, christian, heiner}@informatik.uni-mannheim.de

ABSTRACT

We propose an approach for semantifying web extracted facts. In particular, we map subject and object terms of these facts to instances; and relational phrases to object properties defined in a target knowledge base. By doing this we resolve the ambiguity inherent in the web extracted facts, while simultaneously enriching the target knowledge base with a significant number of new assertions. In this paper, we focus on the mapping of the relational phrases in the context of the overall workflow. Furthermore, in an open extraction setting identical semantic relationships can be represented by different surface forms, making it necessary to group these surface forms together. To solve this problem we propose the use of markov clustering. In this work we present a complete, ontology independent, generalized workflow which we evaluate on facts extracted by NELL and REVERB. Our target knowledge base is DBPEDIA. Our evaluation shows promising results in terms of producing highly precise facts. Moreover, the results indicate that the clustering of relational phrases pays off in terms of an improved instance and property mapping.

Categories and Subject Descriptors

I.2.4 [**Knowledge Representation Formalisms and Methods**]: [Representations (procedural and rule-based)]; I.2.3 [**Deduction and Theorem Proving**]: [Uncertainty, fuzzy, and probabilistic reasoning]

Keywords

Data Integration, Markov Clustering, Enriching Knowledge Bases, Probabilistic Inference

1. INTRODUCTION

State-of-the art information extraction systems like NELL [8], REVERB [16] or OLLIE [25] work on web-scale text corpora. They are based on the general paradigm of *open* information extraction (OIE) [3]. Systems that follow this paradigm are not constrained by the boundaries of encyclopedic knowledge or a corresponding fixed schemata. On the contrary, approaches like YAGO [40], DBPEDIA [1] or FREEBASE [4] follow a different paradigm and extract knowledge from partially structured resources. The facts stored in these knowledge bases (KBs) are, unlike extracts by OIE systems, assertions that use an URI scheme to uniquely identify both instances and concepts in its ontology.

In our work, we focus on the task of mapping the facts extracted by an OIE system to a target ontology, DBPEDIA for instance. The results of such a mapping process are precise and unambiguous KB assertions, compared to the raw, ambiguous OIE facts. From the perspective of the target KB, an automated mapping approach allows to extend the KB, or a poorly covered sub-domain of the KB, significantly.

Our approach is not limited to any domain and is applicable to any natural language text extracted from the web having the general form $rel(s, o)$; where rel defines a binary relation between the subject term s and object term o. For instance, consider the following REVERB fact [1]:

is a town in (Croydon, London)

If we want to map this fact to an ontology, like DBPEDIA, the relational phrase might possibly map to `dbo:county`, and the terms should be mapped to `db:Croydon` and `db:London` respectively[2]. The task of pinpointing to the exact instance is difficult since both *Croydon* and *London* are extremely polysemous. *Croydon* refers to 59 different entities and *London* to 441 entities in WIKIPEDIA. Given these respective correct mappings, we can translate the original fact to a KB property assertion as

`dbo:county(db:Croydon, db:London)`

It must be noted that instance and property matching tasks are not independent. If we correctly map "*is a town in*", we know that subject and object term will refer to geographical entities. Inversely, a fact that involves two geographical entities increases the probability that the relational phrase refers to a property relating locations. Our approach derives and leverages such dependencies within the overall task of generating new semantified facts.

In an open extraction scenario, different relational phrases often capture the same meaning and might still refer to the

[1]Croydon is large town to the south of London.
[2]`db`: DBPEDIA instances with the namespace http://dbpedia.org/resource/
`dbo`: defines the namespace http://dbpedia.org/ontology/ and can be used either with concepts or properties. We use this nomenclature in the rest of the paper.

same property in the target ontology. For instance, "*is located in*", or "*is a village in*" can nevertheless be translated into a correct assertion using the more general property. So, it is important to first cluster such phrases [24] instead of computing a mapping for each one individually. A relational phrase that appears only rarely in the data set can thus profit from its cluster membership, leveraging comprehensive statistics available via all members of its cluster.

Our main contributions in this paper include (i) a modularised workflow for solving the general task of mapping OIE facts to a target KB, (ii) proposing markov clustering technique to group OIE relational phrases, and (iii) a feedback based approach to improve the overall quality of the assertions that are finally generated. In Section 2 we discuss some of the major related works from this area, followed by a brief overview of our framework in Section 3. In Section 4 we explain the clustering and the property matching techniques in detail where we propose markov clusters for the clustering problem. Empirical results and analysis are presented in Section 5. Finally we conclude in Section 6, with insights into our approach, limitations and scopes of improvements.

2. RELATED WORK

Matching Instances: The task of instance matching has been dealt with previously under the notion of entity linking. Major work in this area was done by Bunescu and Paşca [7] and Cucerzan [9] who focused on the usage of Wikipedia categories and global contexts, respectively. Systems like DBpedia Spotlight [26], AIDA [21], exploit mainly context of the entities. However in our problem setting. context is usually missing within the facts generated by OIE systems, making the task bit harder. In our approach, we apply a method that uses Wikipedia anchor text as surface forms as introduced by Bunescu and Paşca [7]. This allows us to derive the probability of a candidate based on the frequency associated to this surface form.

Knowledge Base Constructions and Debugging: There has been some development towards scalable knowledge base creation with a minimal amount of human intervention. Chen et.al. [44] introduced a system called ProbKB, performing deductive reasoning on web extracted facts by a set of first order logic rules and solving as an inference task in MLN. As a follow up work [43], ProbKB was used for automated KB construction. Our work does not target creation but rather using the open information for extending an already existing structured KB. Also, considerable work has explored unsupervised methods for tasks like acquisition of binary relations [5], facts [14], and instances [30]. Pujara et. al. [32] have used probabilistic soft logic to detect inconsistencies in knowledge graphs by exploiting dependencies within the graph. These works aim at reasoning within a given structure, provided, for example, by the concept and property hierarchy of NELL. In our work, the instance mapping module exploits OIE reasoning to a considerable degree in the context of a target KB like DBpedia and Yago to disambiguate OIE terms. Our assumption is that the source facts might be unstructured and do not maintain a concept/role hierarchy. This makes our approach independent of the existence of a source ontology. Moreover, we do not aim at refining the OIE itself, but use the OIE data in its given form to generate semantified facts.

Distant Supervision based Approaches: On a different note, there has been a lot of work on distant supervision based approaches since the early 90s. Work like DIPRE [6] was first of its kind to use a set of seed KB facts to discover patterns all across the web. Those patterns were used to discover more facts, furthermore, bootstrap these two to learn more facts and more patterns. The idea was also seen in systems like SOFIE [38], where natural language texts were excavated for entities and relationship patterns and most likely entity references were solved as a joined satisfiabilty problem. In particular, systems like PATTY [28] provide a taxonomy of relations. This is yet another example of a system which exploits relational patterns between entities and uses them to create a hierarchy of relational phrases. The authors of PATTY tried to paraphrase DBpedia and Yago entity relations with multiple paraphrases. This is different, since our approach tries to find a mapping given a set of paraphrases. Even Nell [8] and Reverb [17, 15] bear a similar architecture in identifying and finding relationships across the web. More recently, there has been clustering based works by Moro et.al [27], Sun et.al. [41]. This genre of work primarily focuses on the relations from open domain extractions; it extracts them, clusters them and finally disambiguates them. They exploit distributional semantics of relational phrases to aid their clustering (which is based on shortest path kernel method). Eventually, their goal is to disambiguate OIE facts based on the context, for instance "*is part of*" may be used in the sense of a location part of a larger place, or a person part of a band or organization. However, we have a different objective. We want to fully semantify an OIE triple in terms of a target KB, i.e., we want to select the correct property from the KB given an ambiguous relational phrase (and we have to solve a similar mapping task for subject and object terms). In this context, we should also mention the work by Augenstein et. al [2] cater to the classical problem of relation extraction from web texts and attempts to improve upon state of the art methods.

Semantifying Open Information: Soderland et.al. [36] worked on ontologizing OIE, by both mapping subject and object terms that occur Reverb facts to WordNet, additionally learning relations, such as entailment, between property phrases. This allows to derive a property taxonomy expressed in a normalized vocabulary of the OIE system, while the object and subject terms are mapped to an existing target vocabulary. It is thus partially similar to our approach, however, the mapping of relational phrases is not covered. Moreover, this approach has only been applied to the domain of health and nutrition. We found close resemblance of our work to the work of Soderland et.al [37], which aims at mapping the relational phrases from Reverb to "domain relations". They adopt a two step approach: first, by finding a suitable class recognizer for the OIE terms, second, learning a mapping of the relation under the constraints of the classes recognized. However, they used NFL (National Football League) as the target KB, unlike broader DBpedia in our case. The concept of a "Universal Schema" as introduced by [35] is interesting as they solve the task by defining an union of schemas from source inputs, in an attempt to fix incompleteness generated in mapping to target schemas. They used Freebase as the target KB and use matrix factorization to rectify both structured and unstructured data. However, we exploit the unstructured sources to enrich the structured KB. PARIS [39] is another well known tool which needs to be mentioned in this context. It performs proba-

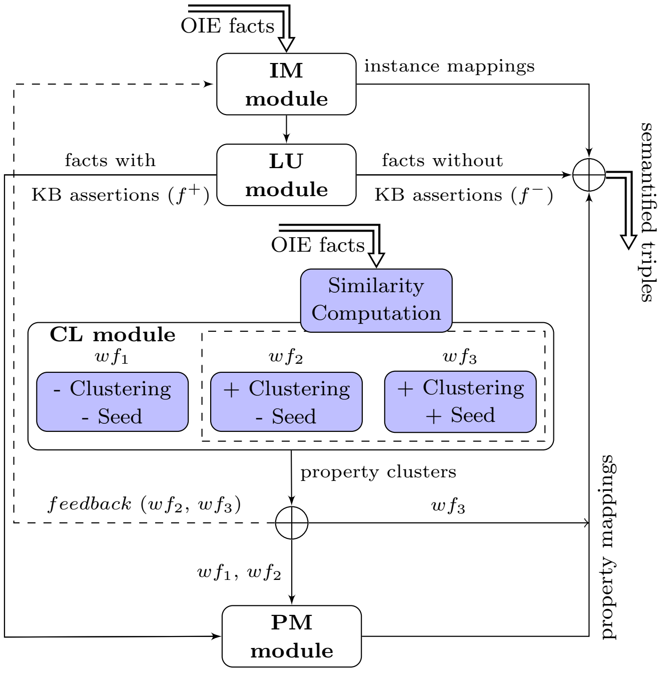

Figure 1: The system architecture.

bilistic alignment of relations, instances and schema across ontologies. This work focuses particularly on aligning two ontologies, for instance the authors of PARIS matched YAGO and DBPEDIA. However, we align an unstructured source to a structured target ontology. The key aspect of our methodology is the assumption that source is without an ontology and thereby making our approach applicable to any kind of source input. More recently, work by Presutti et. al [31] discovers semantic relations from inter wiki page links. However, they have a very simplistic edit distance based entity matching scheme and they exploit the available sentence to create a RDF graph representation. We have a more general framework capable of working without context.

3. OVERVIEW

In Figure 1, we depict the framework for translating OIE facts to assertions in the target knowledge base. It consists of four primary interacting components.

The **instance matching (IM)** module takes as input facts from OIE systems. Additionally, it may also accept groups of OIE facts, clustered together based on their relational phrases. The aim is to find the correct or most probable KB reference to every subject and object term in a REVERB fact. The outcome of the IM module is a mapping of the subject and object terms to DBPEDIA instances. This module is not the core contribution of the paper, but we present its working mechanism in brief. Every REVERB instance term (not the relational phrase) can be considered as a surface form representation of the actual entity in context. For instance, "*london*" might refer to the cricket stadium "The Oval", the capital city of UK, or even the poem by William Blake. Hence, there is a considerable amount of ambiguity involved in finding the correct entity reference. We exploited the English Wikipedia and looked for all the anchor texts along with their respective directed article links. We used Wikiprep [18, 19] for the pre-processing, resulting in tuples like <*anchor, article, #link*>, for instance <London, University of London, 118> or <London, London, 121299>. From such a collection of tuples, it is easy to get a probabilistic ranking of the possible referred entities. This simple but robust approach originates from [7], was applied to mapping OIE facts in [13], and has further been used in [11]. The algorithm considers top-k possible DBPEDIA candidates and exploits, for a particular relation pattern like "*is a suburb of* $(*, *)$", a probabilistic distribution over the likely domain/range. For this pattern, `dbo:City` is more likely to be a domain than `dbo:Scientist`, and so for the range. A more likely domain/range restriction enhances the candidate matching. This mutual reinforcement cycle is embedded in Markov logic network [34], and the improbable mappings are filtered out by finding a MAP[3] state of the network, where a maximum number of correct mappings hold true under a set of constraints.

The **look up (LU)** module searches for facts that are already stated in the target KB, DBPEDIA in this case. The input to this module is a set of instance mappings generated by IM module. Particularly, for each OIE fact, f, if the subject maps to x and object to y, we search for assertions in DBPEDIA that relate x and y. Multiple property assertions involving x and y is also likely[4]. Some of the assertions in this set might correspond to the given OIE fact. These are called f^+; facts with KB assertions and are fed into the PM module as evidences for mapping, which will be explained later on. If the set is empty, the facts are classified as f^-; facts without KB assertions. These facts are, at the end of the overall workflow, translated into the DBPEDIA vocabulary. The idea for this module is to separate OIE facts which can be used for knowledge base extension(f^-) from those which can serve as evidence(f^+). This is especially important since we do not intend to generate a KB assertion like `dbo:country(db:London, db:United_Kingdom`, which already exists in the KB.

The **clustering (CL)** module generates as output clusters of relational phrases having similar meaning, i.e., that can be correctly mapped to the same property. We implemented three different clustering methods including the trivial case for which we treat each relational phrase as a one element cluster. There are two non-trivial clustering methods. One works with DBPEDIA input seeds and the other one works without DBPEDIA seeds. Both methods are explained in detail in Section 4.3. These two methods require a similarity score computed for each pair of relational phrases in the input OIE data set. The similarity computation module is hence applied only on these two non-trivial clustering methods. As illustrated in Figure 1, we execute three different workflows wf_1, wf_2 and wf_3 that differ with respect to the applied clustering technique. The output of this module is clusters of relational phrases, which includes the simplest case where the phrases constitute a one element cluster, i.e for wf_1. The clusters generated by wf_2 and wf_3 are forwarded to the IM module (dashed arrow in Figure 1), which is executed again to improve the instance mapping due to better statistical coverage of clusters compared to the coverage of individual relational phrases.

[3]Maximum a Posteriori

[4]`dbo:location(db:London, db:United_Kingdom)` `dbo:capital(db:London, db:United_Kingdom)`

The **property mapping (PM)** module tries to map a relation phrase or clusters of such phrases to a target KB object property. It must be noted, the PM module tries to map OIE properties to KB object properties[5]. The underlying mechanism for this module is an association rule mining based approach, which attempts to mine for the frequent rule pattern of the form $rel \rightarrow (domain, range)$ (described in [12]). Observe that the output f^+ from the LU module is also an input, serving as evidences for the association rules formation. Every OIE fact which is also "observed" in the target KB in some relational form, can be considered to be a strong evidence for a likely mapping. For example, consider the OIE fact "'*is the voice of(Glenn Tilbrook, Squeeze)*" the analogous fact we observed in DBpedia was "`dbo:associatedBand(db:Glenn_Tilbrook, db:Squeeze_(band))`". This provides a possibility that "*is the voice of*" might be mapped to "`dbo:associatedBand`". The rule we generate is *is the voice of* $\rightarrow$ (`dbo:MusicalArtist`, `dbo:Band`)

In general, the whole set of f^+ facts provides evidences for a possible mapping. This module exploits the domain and range of a DBpedia property in the stated assertion. The first workflow wf_1 involves a direct application of this technique on Reverb. We extend the algorithm for workflow wf_2 to treat clusters of relational phrases as well. Eventually, this module outputs a set of property mappings. Note that clustering with DBpedia properties as seeds (wf_3) implicitly solves the property mapping problem, since each relational phrase is mapped to the DBpedia seed in that cluster. Thus, the PM module is not used by wf_3.

Given the instance and property mappings for a certain fact, each component of the fact can be translated directly to an assertion formulated in the vocabulary of the target knowledge base. Since we apply this translation to f^- only, we know that each generated assertion is completely new. For instance, we started from the OIE fact "*originated in(Japanese honeysuckle, Japan)*", and generated the new assertion "`dbo:origin(db:Lonicera_japonica, db:Japan)`", with the guarantee that there are no pre-existing assertions in DBpedia with some other object property.

4. CLUSTERING AND MAPPING

In this section, we focus on the methodology for solving the clustering of relational phrases and the resulting mapping task. Often, the same relationship might be expressed by the use of different relational phrases. For instance, phrases like *"is the partner of", "was the spouse of", "is married to"* denote the same relationship. Hence, it is rational to group these phrases into a cluster which can then be treated as one collective unit.

For computing these clusters, we apply a graph based approach in which nodes represent relational phrases and edges weigh the degree of similarity of the connected phrases. We first explain how to compute the pairwise similarity in Section 4.1. In Section 4.2 we introduce the clustering techniques that we apply on the undirected edge-weighted graphs. Finally, we explain the property mapping technique in Section 4.3.

[5]properties defined between entities in the KB, for instance, `dbo:author` which is defined for `dbo:Work` and `dbo:Person`. This is different from data type properties which have literals as objects/subjects, like `dbo:foundationDate` defined between `dbo:PopulatedPlace` and `xsd:date`

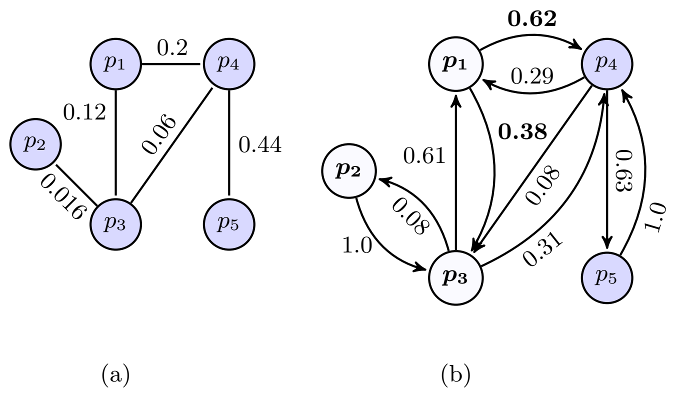

Figure 2: (a): A weighted undirected graph representing similarities between nodes. (b): same graph showing the transition probabilities. The directed edges represent the probability of moving to the connecting node. Note that the bold values add up to 1. p_1**:** ***is a village in*****;** p_2**:** ***is a town in*****;** p_3**:** ***is a suburb of*****;** p_4**:** ***currently resides in*****;** p_5**:** ***currently lives in*****; nodes of same color are eventually in the same cluster.**

4.1 Similarity Metrics

We introduce a similarity measure $sim(r_i, r_j)$ which captures the degree of similarity between any two property phrases $r_i, r_j (i \neq j)$. The measure is defined as,

$$sim(r_i, r_j) = \beta * jac(r_i, r_j) + (1 - \beta) * wn(r_i, r_j)$$

where, $jac(r_i, r_j)$ denotes a jaccard similarity and $wn(r_i, r_j)$ denotes a WordNet similarity, and *beta*, β, is a weighing factor ($0 \leq \beta \leq 1$). All similarity scores are normalized between 0 and 1. In Section 5.3.2, we present an empirical analysis for our choice of β.

The jaccard similarity $jac(r_i, r_j)$ is defined as follows. Let f_{r_i} and f_{r_j} be the set of facts having the relational phrase r_i and r_j respectively. If n denotes the number of instance pairs where both the subject and object terms are in common across both sets, then the jaccard similarity is, in our context, defined by, $jac(r_i, r_j) = n/|f_{r_i} \cup f_{r_i}|$. For computing the WordNet score we used a similarity API[6] which internally uses a hybrid approach involving statistics from a large corpus [23, 20]. We added jaccard co-efficient, which has been proven effective in document clustering[22, 29], to incorporate a component involving the data set characteristics. Applying the measure to the phrases r_i = *is the capital of* and r_j = *is the capital city of*, for example, we obtain the score 0.505 with $wn(r_i, r_j) = 0.829$ and $jac(r_i, r_j) = 0.181$ if we set $\beta = 0.5$.

4.2 Markov Clustering

We apply markov clustering (MCL) to generate the clusters that we finally map to the properties in the target ontology. MCL works well with large inputs and it has been applied successfully for finding semantically similar words [10]. The primary work on MCL was done by van Dongen [42]. In Figure 2(a), we depict an example for a set of five Reverb properties p_1 to p_5. The nodes represent individual property phrases. A weighted edge connecting

[6]http://swoogle.umbc.edu/SimService/

two nodes denotes their affinity score (introduced as *sim* in Section 4.1). A missing edge denotes absolute dissimilarity, for instance, there exists no edge between p_5 and p_3. These weights are converted to probabilities. For instance, node p_1, is connected to p_3 and p_4, hence the transition probability to p_3 is given by $0.12/(0.12 + 0.2) = 0.38$ as shown in Figure 2(b) with a directed edge. The sum of all the transition probabilities from a particular node has to be 1.0.

A graph with n nodes will have a probability matrix, $\mathcal{P} \in \mathbb{R}^{nXn}$. An element $p_{ij} \in \mathcal{P}$ represents the probability of node j transiting to node i. And so essentially

$$\sum_{i=1}^{n} p_{ij} = 1.0$$

thereby making $\mathcal{P}$ a column stochastic matrix. Donjen et. al. have simulated the concept of random walk using markov chains, where the steady state probability distribution is obtained by an iterative product of the probability matrix. Hence the n^{th} state probability is given by $\mathcal{P}^n$. The idea is to make a stronger link even stronger and weaker ones even weaker, with these repetitive steps. Computing matrix products and re-normalizing them is called *inflation* and is controlled by the parameter *inflation factor*, I. We used the available implementation[7] for our experiments.

There are two major challenges in applying the method successfully. First, the choice of the inflation factor I is crucial. Setting it too small, makes the cluster too coarse and vice versa. And second, our initial experiments showed that with a reasonable I, some of the final clusters had further local sub-clusters. We implemented an extension to the algorithm which allows to find these local sub-clusters. For a given I, we apply markov clustering on each of the cluster, ignoring the influences from all other clusters. In Section 5.3.2, we report about experiments to automatically determine the optimal choice of I, which avoids a random selection of this parameter.

4.3 Property Mapping Strategies

We aim at mapping REVERB relational phrases or clusters of such phrases to DBPEDIA object properties. The abstract algorithm for the complete framework is presented in Algorithm 1. It accepts as input the different workflow modes and performs the mapping task accordingly. In workflow wf_1, we treat each REVERB relational phrase individually and map it to a DBPEDIA property. This involves a direct application of the rule based approach described in [12].

The second workflow wf_2, is an extension of the former, but involves clustering (Algorithm 1, line 14) the REVERB relational phrases. Once the relational phrases are clustered, we apply the rule based technique on the clusters. Here we treat each cluster as a single property for which the set of associated facts is the union of all facts containing one of the relational phrases in the cluster. The output of this workflow is a set of clusters, where each cluster is mapped to a DBPEDIA property. Each phrase in a cluster is eventually mapped to the cluster specific DBPEDIA property. For instance, "*is a city in*" and "*is a village in*", belonging to same cluster, are mapped both to `dbo:isPartOf`.

In workflow wf_3, we opt for a purely cluster based approach in which the mapping task is automatically solved by computing the clusters. Here, we *add* DBPEDIA properties

Algorithm 1 Algorithm for Semantfying Web Facts

```
Require: $F$: facts from OIE system; $mode$
 1: function GENFACTS
 2:     $i_{maps} \leftarrow null$          ▷ instance mappings collection
 3:     $p_{maps} \leftarrow null$          ▷ property mappings collection
 4:     $phrases \leftarrow relational\ phrases\ from\ F$
 5:     $cl \leftarrow cluster(phrases)$          ▷ call to line 14
 6:     $i_{maps} \leftarrow IM(cl)$          ▷ call to IM module
 7:     $f^-$, $f^+$ using $LM(i_{maps})$          ▷ call to LU module
 8:     if $mode = wf_1\ or\ wf_2$ then
 9:         $p_{maps} \leftarrow PM(f^+)$          ▷ call to PM module
10:     else
11:         $p_{maps} \leftarrow from\ cl$
12:     $newFacts \leftarrow combine\ i_{maps}, p_{maps}, and\ f^-$
13:     return $newFacts$

14: function CLUSTER(p)
15:     $c \leftarrow null$
16:     if $mode = wf_1$ then
17:         $c \leftarrow one\ element\ cluster$
18:     else          ▷ $wf_2$ and $wf_3$
19:         if $mode = wf_3$ then
20:             $p \leftarrow p$+DBPEDIA $seeds$
21:         $simCompute(P)$          ▷ similarity scores
22:         $c \leftarrow markov\ cluster\ on\ p$
23:     return $c$
```

as seeds and allow them to be clustered with the REVERB phrases using the markov clustering technique. Pairwise similarity scores (Algorithm 1, line 21) between the newly fed DBPEDIA properties and the REVERB relational phrases need to be computed in that setting. Applying clustering on this heterogeneous mixture of REVERB phrases and DBPEDIA properties results in clusters where most of the clusters had a DBPEDIA property along with a set of other REVERB properties. For instance, {`dbo:origin`, "*originated in*", "*is the source of*", "*is a source of*"} is one of the clusters. However, the major drawback of this approach is that, we cannot ensure that every final cluster contains a DBPEDIA property. There can be clusters without any DBPEDIA property, for instance {*"is ideal for", "is perfect for"*}. This might be the effect of an unsuccessful clustering or, DBPEDIA might not have an analogous property capturing the similar sense of a cluster of REVERB phrases.

5. EXPERIMENTS

In the following we present our experiments and discuss their results. In Section 5.1 we describe the used datasets. In Section 5.2 we refer to the procedure we applied to evaluate the results of our approach. In Section 5.3 we explain how we searched for the optimal parameter setting. Finally, we present and discuss the results in Section 5.4.

5.1 Experimental Settings

We used the latest REVERB ClueWeb Extractions dataset[8], consisting of approximately 15 million facts annotated with a confidence score. Unlike the smaller REVERB Wikipedia extractions data set (0.4 million), ClueWeb is larger and contains more domain independent facts, thereby providing

[7]http://www.micans.org/mcl/index.html?sec_software

[8]reverb.cs.washington.edu/reverb_clueweb_tuples-1.1.txt.gz

REVERB fact	DBPEDIA assertion	fact clear?	subject correct?	property correct?	object correct?
is a kind of(Croatia, country)	`dbo:type(db:Croatia, db:List_of_sovereign_states)`	✓	✓	✓	×
is located in(Yekaterinburg, Ural Federal District)	`dbo:region(db:Yekaterinburg, db:Ural_Federal_District)`	✓	✓	✓	✓
could not be reached for(Jacob, comment	`dbo:knownFor(db:Jacob, db:Comment_(computer_programming)`	×	-	-	-

Table 1: Sample annotation schemes. (✓= correct; × = incorrect; - = not annotated)

a generalized setting for our approach. For our experiments we considered only triples with at least 95% confidence and removed facts that contain numeric expressions, (e.g. *saw the start of(1609, ceasefire)*, *were spotted in(161 tornadoes, last month)*), since we are only interested in facts that can be mapped to object property assertions. The filtered dataset contains 3.5 million triples with 474325 different relational phrases. We selected the facts that use the 500 most frequent property phrases, which consisted of 1072215 facts, i.e., approximately 31% of the filtered dataset. It was quite interesting to observe that selecting the top 0.1% relational phrases (500 out of 474325) covers nearly one-third of all facts. As target knowledge base we used DBPEDIA (Version 3.9). With respect to wf_3, we used the most frequent 100 object properties of DBPEDIA as input to the clustering.

As already mentioned, we applied the algorithm of [11] for the instance matching task. This algorithm decides between different matching candidates based on their types. Since some DBPEDIA instances do not have a specified type in DBPEDIA we additionally exploited the type information available in YAGO. As experimentally verified with cross-validation in [11] (Section Experiments: Learning α), we set α to 0.5, the only parameter of the IM module.

Note that we include in our results presentation numbers reported in [12]. These results are based on running workflow wf_1 on a NELL dataset. NELL uses already a controlled set of distinct relations in its facts. Thus, the clustering problem does not exist. It is thus interesting to see whether it is possible to achieve similar results for the harder REVERB problem scenario, where we cluster property phrases in the context of our overall workflow.

All experiments were conducted on a 64-bit Linux machine with 4GB physical memory and 2 cores. However, for the similarity computation module, we implemented it as an asynchronous and multi-threaded application, allowing us to exploit the multiple cores of a 8-core machine.

5.2 Evaluation

A random sample of 500 new facts was extracted from the final output of each workflow and annotated. The first question we ask to the annotator is, if the whole of REVERB fact is fully comprehensible. If so, only then it is possible to determine the correctness of the candidate mappings. Marking a fact as ambiguous is important since we do not consider those facts for evaluation. Since for such facts the annotator was unable to determine the correct references to the terms in the first place and hence a mapping, if generated, would still remain dubious about its correctness. One such fact is the third example of Table 1 where it is impossible to determine "*which Jacob?*" the fact is talking about.

If the triple is decipherable, we ask the subsequent set of questions: Is the mapping of the subject term, the mapping of the object term, and the mapping of the relational phrase correct in the given context of the fact? Based on these annotations, a fully mapped REVERB fact was considered to be translated correctly, if all the three questions were answered positively. If one of the mappings is annotated as incorrect, the whole translation was considered wrong. This can be seen with the first example in Table 1, where a wrong mapping for *country* makes the whole translation wrong, while the second fact is considered to be a correct translation. Likewise the "-" marks denote that the mappings were not annotated on account of the REVERB fact being ambiguous. Note that, there are no ambiguous annotations, there are only ambiguous input REVERB triples.

The annotation result allows us to compute approximated precision scores based on the evaluated sample. Assuming our method outputs only the three facts presented in Table 1, instance precision will be 3/4 or 75%, since one of the instance matching (*country* $\rightarrow$ `db:List_of_sovereign_states`) was wrong out of the 4 mappings. Similarly, the property precision will be, 2/2 or 100% and fact precision will be 1/2 or 50%. Note, that the ambiguous fact was not considered for evaluation.

5.3 Cluster Analysis

5.3.1 Metric

A substantial part of this work deals with different clustering strategies. We had two parameters in our overall approach, I and β, that have probably an effect on the cluster quality. To quantify this, we define a cluster quality score, which considers two factors, intra-cluster and inter-cluster sparseness. For a set of clusters, $C = \{c_1, \ldots, c_{|C|}\}$, we measure the cluster outputs in terms of a quality measure [33], denoted by S and defined as,

$$S = \left(\sum_{c_i \in \mathcal{C}} \frac{comp(c_i)}{iso(C)}\right)^{-1}$$

where, $comp(c_i)$ denotes compactness and is defined as

$$comp(c_i) = min(sim(r_i, r_j)); \forall r_i, r_j \in c_i$$

It measures how tightly any two arbitrary phrases r_i and r_j are connected in cluster c_i by looking at the minimum pairwise score between all elements in c_i. Note that $comp(c_i)$ is defined only if a cluster has at least two elements. Otherwise, we set it to zero. The metric $iso(\mathcal{C})$ measures isolation. It is defined as

$$iso(\mathcal{C}) = max(sim(r_i, r_j)); \forall r_i \in c_i; \forall r_j \in c_j; c_i \neq c_j$$

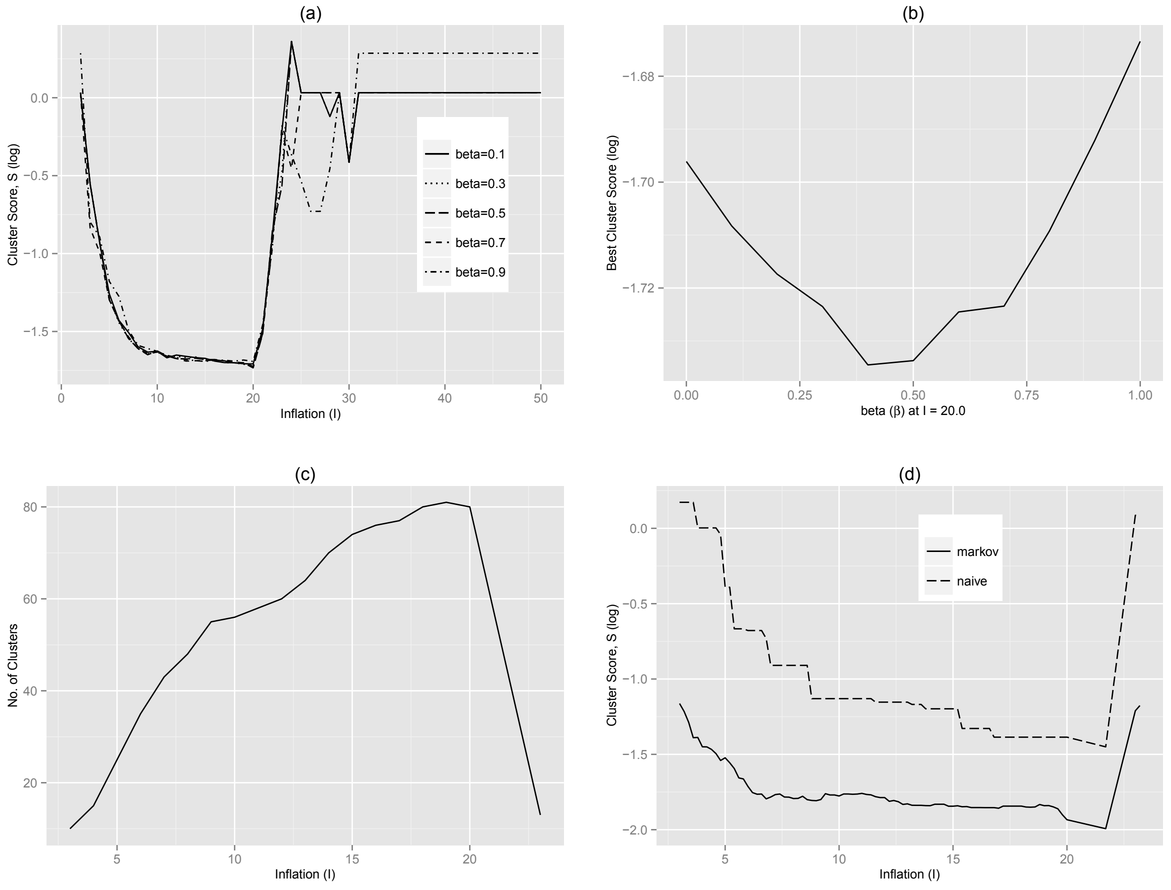

Figure 3: (a) Cluster scores for the range of I, for varying values of β (b) Best cluster scores (minimum score) for various β at I=20 (c) Number of clusters generated depending on I (d) Comparison of the Markov clustering based scheme with a naive mediod based scheme (N.B. All the cluster scores are presented in log10 scale, for making the finer differences more prominent)

It denotes how sparsely the elements are distributed across clusters in C. Ideally, for a good cluster scheme, every cluster c_i should contain very similarly elements i.e high compactness and there should very low similarity between elements across different clusters, i.e. low isolation. This tends to make S low for good clustering schemes.

5.3.2 Parameter Search

In this section we present a principled way of choosing the optimal parameters for our experimental settings. In this initial setup, we alter β in steps of 0.1 starting from 0 to 1.0. For each of these settings, we obtain different pairwise scores for our set of top-500 REVERB relational phrases. For every given β, we alter the inflation factor I (introduced in Section 4.2) in steps of 1.0 ranging from 2.0 to 50.0. We started by setting this factor to 1.0, but the clustering process did not terminate. The choice of the upper limit is not strict but empirical. It was chosen high enough to observe the general trend over a large range of values. We ran the clustering 11 * 49 times (11 data points for $0 \leq \beta \leq 1.0$; 49 data points for I) and we present the results in Figure 3(a). For clarity, we omitted some of the intermittent values for β from the figure. From the notion of quality, we must remember, the lower the score, better is the cluster quality. In general, we observed a similar pattern across different beta values. This can be generalized into trend sections as follows: a sharp improvement in the cluster quality ($\approx 2 \leq I \leq 10$), a marginal improvement phase ($\approx 10 \leq I \leq 20$), followed by a sharp deterioration and some randomness ($\approx I \geq 20$) and eventually saturation ($\approx I \geq 30$). Furthermore, Figure 3(a) also shows that irrespective of the β, at $I = 20$ we have the best cluster score.

We are interested in a score of β, which would be optimal for the given set of REVERB relational phrases. However, in an attempt to present the pattern over a wide range of values, Figure 3(a) fails to capture the exact β value (at I=20.0) which results in the lowest cluster score and hence the best cluster configuration. For this reason we *zoom-in* the results from Figure 3(a) at I=20, and investigate them further for all the β values and their corresponding cluster score. We present them in Figure 3(b). We achieved the best cluster score for $\beta = 0.4$. This was a two step approach to find the two parameters I and β, by deciding the former first, fixing it and and then deciding the later. In all the subsequent experiments, we used these parameter values.

	NELL	REVERB		
	wf_1	wf_1	wf_2	wf_3
matched rel. phrases	22	26	29	212
matched KB properties	25	52	29	41
new facts generated	1455	59,526	48,212	78,085
instance precision	97%	86.48%	89.88%	95.12%
property precision	96%	59.46%	75.28%	86.18%
fact precision	77%	43.24%	57.30%	78.05%

Table 2: Comparison of workflows.

5.3.3 Comparison with Naive Clustering

We compared the performance of the Markov cluster approach with a variant of k-means clustering. We refer to this method as naive clustering. It selects k random relational phrases from an input set (in our case, $k < 500$) and tries to assign the rest (i.e. 500-k) of the phrases closest to one of these k feed relational phrases. Note that, for the naive clustering, there is no direct influence of I. However, we have chosen the number of clusters generated by the Markov technique, and feed it as k for the naive clustering. Thus we ensure that the number of clusters generated by the two schemes is comparable. In Figure 3(c) we report about the change in the number of clusters generated with an increasing inflation factor. We observe that after a point, there is a sharp drop, denoting that higher I does not necessarily generate higher number of clusters.

However, higher number of clusters does not guarantee a good cluster. Hence, we measure the score S, as introduced above, for capturing the individual qualities of the two schemes. These behaviors are reported in Figure 3(d), where we compare both schemes by plotting I vs S. Since the naive approach is based on random feeds, we perform a repeated random sampling ($\approx$1000 times) for each step of I, and get the best score from them. As discussed in Section 4.2, we try to reduce the granularity of clusters by iteratively re-clustering. For this step, we try to find sub-clusters for clusters with more than 10 elements. While doing this, I was kept the same. It often happened that a sub-division was not possible and the iteration was terminated. The cluster sizes and quality scores reported here are the numbers measured after performing this refinement step. We had an average cluster size of 5.204 at the optimal inflation.

It must be observed that for all values of I, the Markov scheme generates always lower scores than the naive approach. And more interestingly, after the threshold (at I=20) both values tend to go up for both schemes, denoting deterioration in quality. The purpose of these preliminary experiments was to measure the effectiveness of the Markov clustering technique compared to the naive approach. In the following experiments we adopt Markov clustering and run it with different sets of inputs; only on REVERB in wf_2 and on a mixture of REVERB and DBPEDIA in wf_3.

5.4 Main Results

Table 2 summarizes the results for all workflows discussed in the paper. The table header denotes the following:

1. wf_1: considering every relational phrase uniquely, i.e. without any clustering, and applying association rule mining techniques for property mapping
2. wf_2: clustering relational phrases and applying rule mining for mapping the resulting clusters
3. wf_3: clustering relational phrases with DBPEDIA object properties as seeds without applying any rule mining

We compare these different workflows based on the following aspects:

1. *matched relational phrases*: the number of unique OIE relations that have been mapped
2. *matched KB properties*: unique target KB properties to which relational phrases have been mapped, in this case DBPEDIA object properties.
3. *new facts generated*: number of new KB assertions generated
4. *instance precision*: for each completely translated OIE fact, this denotes the fraction of subject and object mappings actually correct
5. *property precision*: similar as above, but fraction of relational phrase mappings actually correct
6. *fact precision*: fraction of translated KB assertions actually correct. A KB assertion is actually correct if all of the subject, object and relational phrases are mapped correctly

For a detailed comparison, we also present the numbers obtained for the NELL data set under the column header "NELL (wf_1)". These are the results as reported by Dutta et al. [12]. As the first step, we ran the full workflow under the simplistic scenario, i.e. wf_1. This is the trivial setting where the rule based mapping technique was applied on the REVERB data. Given that, REVERB is few magnitudes larger than the NELL data set, it was not surprising to observe a greater number of generated facts while applying wf_1 on REVERB. However, the precision suffered heavily. Overall, fact precision was 43.24%. This was mainly caused by a low property precision. This indicates the drawbacks of a rule mining based approach to find a correct mapping for the relational phrases. Note that, in this setting, every relational phrase was reasoned and dealt with individually, completely disregarding the effect of its synonymous sibling phrases. Furthermore, out of top-500 relational phrases, we could match only 26. These 26 relational phrases have been mapped to 52 different DBPEDIA object properties. Often, it was the case that a relation was mapped to almost two or more ($\approx$ 52/26) target KB properties. And during the annotation process, it was found that one or more options were wrong. For instance, the phrase "*is a city in*" was mapped to the following DBpedia properties: `dbo:councilArea`, `dbo:district`, `dbo:isPartOf`, `dbo:leade-rName`, `dbo:timeZone` and more. Clearly, `dbo:leaderName` and `dbo:timeZone` are wrong, and it made the whole translated KB assertion incorrect.

The next improvement was to jointly deal the relational phrases. We clustered the top-500 phrases from Reverb and performed the instance mapping and property mapping using the rule mining technique. With this workflow (wf_2), we expected that the clustering would push the instance matching precision higher due to the feedback loop we incorporated. Our expectation was also to observe a positive impact on the property mapping phase. Results were interesting in this setting, since we achieved a marginal improvement in the instance mapping and quite a lot in the property mapping. This speaks in favor of the clustering approach. We further analyzed the complete set of output mappings for the same relation phrase as we did in wf_1, i.e. "*is a city in*". It had no incorrect mappings to `dbo:leaderName` or `dbo:timeZone`, instead had more valid mappings generated like: `dbo:ceremonialCounty`, `dbo:principalArea` and so on. This is also reflected in the numbers: the number of matched KB properties, reduced from 52 to 29, and precision was raised to almost 75%. Combined with a better instance and property precision, we could achieve much better fact precision of 57% compared to wf_1. Clustering looked promising, but still we were suffering from low recall in the sense that hardly any of the phrases were actually mapped, given that we started with 500 of them.

Finally, we choose the Markov clustering approach but with DBpedia properties as cluster seeds (workflow wf_3). We also re-run the instance mapping module using a feedback loop. We observe an increase in instance mapping precision compared to wf_2. We also observe that the property mapping is much better compared to all previous workflows. The improved instance mapping precision combined with a better property mapping precision leads to a higher number of new facts which are of the same quality as that achieved with Nell. Adding DBpedia properties as seed also helped to achieve better instance mappings. We had higher numbers both in terms of quality and quantity. Significantly, a higher number of different relational phrases were matched, resulting in almost 78K new KB assertions. Furthermore, the number of different target properties is increased. Thus, wf_3 is not only the workflow with the highest precision, but also the workflow that achieves the highest recall values. Analyzing further, the phrase "*is a city in*" was matched to `dbo:city`, `dbo:county`, `dbo:region`, `dbo:state`, Since we considered top-100 DBpedia properties, entries like `dbo:principalArea` or `dbo:canton` were absent. Even more interesting was to observe that `dbo:city` was actually a mapping for *is a city in*, *is a city of*, *is a suburb of*, *is a region in*, *is a town in*, *is the capital city of* and also *is the county seat of*. A large cluster of relational phrases was mapped correctly to properties in the target KB. Mathematically, every DBpedia property is mapped to approximately 5 (=212/41) relational phrases. The mappings for the property `dbo:city` are thus a typical example. Similarly, for `dbo:location`, there were 11 Reverb phases mapping to it, including *is located in*, *is situated in* and *lies in*.

Even though with wf_3 we achieved better recall, we observed that there is often a limitation on the number of phrases which could be mapped. We had clusters of phrases, which could not be mapped or actually had no analogous DBpedia property to be mapped to. Out of 500 Reverb phrases, 274 of them were assigned to some cluster that was not mapped to any DBpedia property. Examples are phrases like {*is an essential part of*, *is essential for*, *is essential to*, *is the essence of*, *is vital to*, *are essential for*, *are essential to*}, which were in the same cluster, but had no KB property to be mapped to. In this case our approach is right in not generating any mappings for this cluster, because there exists just no counterpart for this cluster in DBpedia. However, the clustering itself worked fine, since these relational phrases have obviously similar meanings.

Our approach generated also the cluster {*is similar to*, *are similar to*, *is very similar to*}. The analogous DBpedia property is `dbo:similar`, however, it was not in our top-100 seed properties. Including more DBpedia properties can help to map some of these 274 non-mapped relational phrases. We roughly sampled 10 clusters (containing 31 phrases) from the set of output clusters involving these 274 relations. The idea was to get a very rough number about the fraction which can be mapped. We manually investigated each of these samples, and found that out of the 10 clusters, 7 had no analogous DBpedia property, while we might have been able to map 3 if we would had included more DBpedia seeds. Note that in these 10 clusters there was no relational phrase for which a counterpart in the top-100 seeds existed. It seems that our algorithm matched all matchable relational phrases given the restriction of using the top-100 seed of DBpedia properties only.

Note that the fact precision for Nell (wf_1) and Reverb (wf_3) are comparable. This indicates that the clustering approach with DBpedia properties as input seed is better suited for mapping Reverb facts and produces a large number of precise facts. Whereas, the rule mining approach seemed to perform equally well for Nell but Reverb achieved worse performance with that same technique (wf_1). This highlights the importance of applying an appropriate clustering technique within an overall framework that deals with OIE systems like Reverb or similar ones having raw non-normalized textual relational phrases.

6. CONCLUSION AND FUTURE WORK

While these results are promising, we identify few areas which could be improved and highlight some key aspects of our method.

- Our approach considers a Wikipedia-like textual corpus work. But, it is not indispensable. We need a ranked list of entity-candidate mappings as input, irrespective of how that was generated.

- Our framework is robust to KB incompleteness and can easily handle missing types for instances and missing domain/range restrictions for the properties. It is always better to have them but A-box completeness is never an absolute necessity.

- We consider the number of facts generated ($\approx$ 78K) still to be low, given that we started with 1072215 Reverb facts. The generation factor is approximately 7.8% of the input data size. This bar can be raised by selecting a larger Reverb corpus, not just selecting the top-500 relations. Also we are considering additional input sources, from text corpora like news archives.

- For the IM module, it is sometimes difficult to make a choice between the candidates of the same type, for e.g. `db:Vienna,_Maryland` and `db:Vienna,_Illinois`, both of type `dbo:Place`. Kernel based density estimation can help in this scenario.

- We will try to use another target KB like Yago and compare the property phrase mappings generated by our system to those generated by PATTY [28].

- We have created a gold standard and working towards finalizing its release. We also extended our framework to incorporate an evaluation module, developed as an easily plug-able framework to our existing framework. We are planning to release the source code and the framework.

- Our current experiments with complete Reverb and DBpedia properties did not yield any scalability issues. Several internal modules are designed to work on distributed nodes. Only bottleneck is the MAP state computation with RockIt inference engine, used in IM module. It cannot work in distributed setting and can sometimes raise issues with large inputs.

- Our target is to generate A-box knowledge and not T-box. This is something we would like to investigate more, since, often several OIE relations capture interesting relationships which are completely missing in the target KB. Presutti et.al [31] tackles this problem, but on a custom ontology, we would like to define new properties on the target KB available.

We presented a general framework for generating new knowledge from a large corpus of web extracted facts in terms of a target knowledge base vocabulary. We applied our framework to the facts generated by Reverb in order to enrich Dbpedia. In doing so, we analyzed three different workflows. In particular, we used Markov clustering as a means for clustering relational phrases and showed its effectiveness in improving instance matching and property mapping in a bootstrapped way. In our experiments, we showed that best results were achieved by using the target properties of DBpedia as seed for the clustering process. In particular, we were able to generate 78,085 new facts with an estimated precision of 78.05%.

7. ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of a Google Faculty Research Award (see http://dws.informatik.uni-mannheim.de/en/projects/current-projects/). We would also like to extend our gratitude to Dr. Mathias Niepert from University of Washington, for sharing his valuable ideas in the initial phases of this work.

8. REFERENCES

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *Proc. of the 6th International Semantic Web Conference*, pages 722–735, 2007.

[2] I. Augenstein, D. Maynard, and F. Ciravegna. Relation extraction from the web using distant supervision. In *Knowledge Engineering and Knowledge Management - 19th International Conference, EKAW 2014, Linköping, Sweden, November 24-28, 2014. Proceedings*, pages 26–41, 2014.

[3] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the Web. In *Proc. of IJCAI-07*, pages 2670–2676, 2007.

[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proc. of the 2008 ACM SIGMOD international conference on Management of data*, 2008.

[5] S. Brin. Extracting patterns and relations from the World Wide Web. In *Proc. of WebDB Workshop at EDBT-98*, 1998.

[6] S. Brin. Extracting patterns and relations from the world wide web. In *In WebDB Workshop at 6th International Conference on Extending Database Technology, EDBTâĂŹ98*, pages 172–183, 1998.

[7] R. Bunescu and M. Paşca. Using encyclopedic knowledge for named entity disambiguation. In *Proc. of EACL-06*, 2006.

[8] 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.

[9] S. Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *Proc. of EMNLP-CoNLL-07*, 2007.

[10] B. Dorow, D. Widdows, K. Ling, J.-P. Eckmann, D. Sergi, and E. Moses. Using Curvature and Markov Clustering in Graphs for Lexical Acquisition and Word Sense Discrimination. *eprint arXiv:cond-mat/0403693*, Mar. 2004.

[11] A. Dutta, C. Meilicke, and S. P. Ponzetto. A probabilistic approach for integrating heterogeneous knowledge sources. In *ESWC-14*, pages 286–301, 2014.

[12] A. Dutta, C. Meilicke, and H. Stuckenschmidt. Semantifying triples from open information extraction systems. In *Frontiers in Artificial Intelligence and Applications*, volume 264. IOS Press, 2014.

[13] 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.

[14] 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 *WWW*, 2004.

[15] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and M. Mausam. Open information extraction: The second generation. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One*, IJCAI'11, pages 3–10. AAAI Press, 2011.

[16] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proc. of EMNLP-11*, 2011.

[17] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proc. of EMNLP-11*, 2011.

[18] E. Gabrilovich and S. Markovitch. Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge. In *Proc. of AAAI-06*, 2006.

[19] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proc. of IJCAI-07*, 2007.

[20] L. Han, A. Kashyap, T. Finin, J. Mayfield, and J. Weese. Umbc ebiquity-core: Semantic textual similarity systems. *2nd Joint Conf. on Lexical and Computational Semantics, Association for Computational Linguistics*, page 44, 2013.

[21] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. Robust disambiguation of named entities in text. In *Proc. of EMNLP'11*, 2011.

[22] A. Huang. Similarity Measures for Text Document Clustering. In J. Holland, A. Nicholas, and D. Brignoli, editors, *New Zealand Computer Science Research Student Conference*, pages 49–56, Apr. 2008.

[23] A. Kashyap, L. Han, R. Yus, J. Sleeman, T. Satyapanich, S. Gandhi, and T. Finin. Meerkat mafia: Multilingual and cross-level semantic textual similarity systems. *SemEval 2014*, page 416, 2014.

[24] D. Lin and P. Pantel. Dirt @sbt@discovery of inference rules from text. In *Proc. of KDD '01*, KDD '01, pages 323–328. ACM, 2001.

[25] Mausam, M. Schmitz, R. Bart, S. Soderland, and O. Etzioni. Open language learning for information extraction. In *Proc. of (EMNLP-CONLL)*, 2012.

[26] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. DBpedia Spotlight : Shedding light on the web of documents. In *Proc. of I-Semantics'11*, 2011.

[27] A. Moro and R. Navigli. Integrating syntactic and semantic analysis into the open information extraction paradigm. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 2148–2154. AAAI Press, 2013.

[28] N. Nakashole, G. Weikum, and F. M. Suchanek. PATTY: A taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 1135–1145, 2012.

[29] S. Niwattanakul, J. Singthongchai, E. Naenudron, and S. Wanapu. Using of jaccard coefficient for keywords similarity. In *Proceedings of the International MultiConference of Engineers and Computer Scientists 2013 Vol I*, 2013.

[30] 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.

[31] V. Presutti, S. Consoli, A. G. Nuzzolese, D. R. Recupero, A. Gangemi, I. Bannour, and H. Zargayouna. Uncovering the semantics of wikipedia pagelinks. In *Knowledge Engineering and Knowledge Management - 19th International Conference, EKAW 2014, Linköping, Sweden, November 24-28, 2014. Proceedings*, pages 413–428, 2014.

[32] J. Pujara, H. Miao, L. Getoor, and W. Cohen. Large-scale knowledge graph identification using psl. In *AAAI Fall Symposium on Semantics for Big Data*, 2013.

[33] B. Raskutti and C. Leckie. An evaluation of criteria for measuring the quality of clusters. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, IJCAI '99, pages 905–910, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[34] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2), 2006.

[35] S. Riedel, L. Yao, B. M. Marlin, and A. McCallum. Relation extraction with matrix factorization and universal schemas. In *(HLT-NAACL '13)*, 2013.

[36] S. Soderland and B. Mandhani. Moving from textual relations to ontologized relations. In *AAAI Spring Symposium: Machine Reading*, pages 85–90. AAAI, 2007.

[37] S. Soderland, B. Roof, B. Qin, S. Xu, Mausam, and O. Etzioni. Adapting open information extraction to domain-specific relations. *AI Magazine*, 31(3):93–102, 2010.

[38] F. Suchanek, M. Sozio, and G. Weikum. SOFIE: A self-organizing framework for information extraction. In *WWW'09 : proceedings of the 18th International World Wide Web Conference*, pages 631–640, Madrid, Spain, 2009. Association for Computing Machinery (ACM), ACM.

[39] F. M. Suchanek, S. Abiteboul, and P. Senellart. PARIS: Probabilistic Alignment of Relations, Instances, and Schema. *PVLDB*, 5(3):157–168, 2011.

[40] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge. In *Proc. of WWW-07*, 2007.

[41] A. Sun and R. Grishman. Semi-supervised semantic pattern discovery with guidance from unsupervised pattern clusters. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 1194–1202, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[42] S. van Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, Utrecht, May 2000.

[43] D. Z. Wang, Y. Chen, S. Goldberg, C. Grant, and K. Li. Automatic knowledge base construction using probabilistic extraction, deductive reasoning, and human feedback. In *Proceedings of the Joint Workshop on*, AKBC-WEKEX '12, pages 106–110, 2012.

[44] 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*, June 2013.