Canonicalizing Open Knowledge Bases

Luis Galárraga, Geremy Heitz, Kevin Murphy, Fabian M. Suchanek

Canonicalizing Open Knowledge Bases

Luis Galárraga
Télécom ParisTech
luis.galarraga@enst.fr

Geremy Heitz
Google Inc.
geremy@google.com

Kevin Murphy
Google Inc.
kpmurphy@google.com

Fabian Suchanek
Télécom ParisTech
fabian.suchanek@enst.fr

ABSTRACT

Open information extraction approaches have led to the creation of large knowledge bases from the Web. The problem with such methods is that their entities and relations are not canonicalized, leading to redundant and ambiguous facts. For example, they may store ⟨*Barack Obama, was born in, Honolulu*⟩ and ⟨*Obama, place of birth, Honolulu*⟩. In this paper, we present an approach based on machine learning methods that can canonicalize such Open IE triples, by clustering synonymous names and phrases.

We also provide a detailed discussion about the different signals, features and design choices that influence the quality of synonym resolution for noun phrases in Open IE KBs, thus shedding light on the middle ground between "open" and "closed" information extraction systems.

1. INTRODUCTION

Recent advances in information extraction (IE) have led to the creation of large structured knowledge bases (KBs), such as NELL [5], YAGO [29], Freebase [4], DBpedia [1], Knowledge Vault [7], ReVerb [10], etc. These knowledge bases contain millions of entities (such as people, organizations, locations, or movies), and hundreds of millions of facts about them (such as which actor acted in which movie, which city is located in which country, etc.). This information is usually stored in the form of ⟨*subject, predicate, object*⟩ triples – a format known as RDF.

Techniques based on "Open IE" [3, 9], such as ReVerb [10], allow the subjects and objects to be arbitrary noun phrases, and the predicates to be arbitrary verb phrases. For example, we may extract ⟨*D.C., is capital of, United States*⟩ and ⟨*Washington, capital city of, U.S.*⟩. However, it is not clear if these are talking about the same entities, or even about the same predicate. This means that when we query the KB for facts about an entity by one name, we cannot be sure that we get all facts about the entity. Conversely, we cannot be sure that all the facts we get are about the same entity.

CIKM'14, November 3–7, 2014, Shanghai, China.

At the other extreme, techniques based on "closed IE" require that the subjects, predicates and objects are canonical, i.e., that they have unique ids. This is the approach taken by many KBs, such as YAGO, Freebase, DBpedia, and Knowledge Vault. The disadvantage of this approach is that the coverage is much lower than with Open IE. There are also systems such as NELL, in which the predicates are from a closed class, but the entity names are open.

In this paper, we present an approach that can take a large "open" KB, such as produced by ReVerb or NELL, and convert it into a canonicalized form, where entity and relation names are mapped to canonical clusters. More precisely, our contributions are as follows:

- We show how standard clustering techniques, with simple blocking strategies and similarity functions, give surprisingly good results for entity clustering in both NELL and Reverb data.
- We investigate an interesting interaction between the way similarity functions are trained, and the way they are used in blocking techniques.
- We show how to extend the AMIE approach [12], which was previously proposed for mining Horn rules on KBs, to learn high quality clusters of semantically equivalent relations from Open IE triples.

2. RELATED WORK

2.1 Open Information Extraction

Open IE systems extract triples of the form ⟨*subject, predicate, object*⟩ from natural language text. For example, given the sentence "McCain fought hard against Obama, but finally lost the election", an Open IE system will extract two triples, ⟨*McCain, fought against, Obama*⟩ and ⟨*McCain, lost, the election*⟩. Early systems [3, 31] typically restricted the subject and object to noun phrases. These can be named entities, such as *Obama*, but also common noun phrases, such as *the election*. The predicate can be any sequence of words that appear between the two arguments. This basic approach can harvest a huge number of triples from Web corpora. However, it will also extract uninformative triples, such as ⟨*Hamas, claimed, responsibility*⟩ (where it is not clear for what Hamas claimed responsibility).

The ReVerb approach [10] restricted the predicates to lightweight verbal phrases, which greatly improved the precision of the triples. The Ollie approach [28] relaxed this restriction by expanding the syntactic scope of relation phrases to cover a much larger number of relation expressions, and by adding context information such as attribu-

tion and clausal modifiers. Some approaches use dependency parsing [31], or employ hand-crafted patterns on the parse trees. ClausIE [6] can reason on the dependency trees, and thus extract more robust triples.

All of these approaches have in common that their relationships are not canonicalized. The approaches cannot see that *was born in* and *'s birth place is* denote the same semantic relationship. The approach used in the NELL project [5], in contrast, works with a predefined set of relationships. It will extract triples of the form ⟨*NP, relation, NP*⟩, where the *NP* are noun phrases and *relation* is one of the predefined relation names. Thus, NELL is not strictly speaking an Open IE system. Still, it shares many properties of Open IE systems. Most notably, all of the Open IE approaches, and NELL, cannot extract canonical entities. They cannot see that *Barack Obama* and *President Obama* are two names for the same person. We discuss some solutions to this below.

2.2 Linking and clustering entities

One approach to resolving entity names is to try to map them to an existing list of known entities, such as Wikipedia or Freebase. This is known as entity linkage (or "Wikification" if the target KB is Wikipedia). Typically each mention generates a candidate list of entities (based on string matching), and then this list is re-ranked using a machine learned model. There are several kinds of models: some link each entity mention in isolation using local features (e.g., [15]), some jointly link sets of mentions within a page using local features and global context (e.g., [26]), and some jointly link mentions across sets of pages (e.g., [19]).

One problem with the above approaches is that many pages may refer to new entities that are not already in a KB. This is particularly true for tail entities (i.e., ones that are not popular enough to have a Wikipedia entry), and/or for new or emerging entities. The standard approach to this is to allow each mention to either map to an entity on the shortlist, or to a special NIL or OOKB (out-of-KB) entity (see e.g., [17]). However, that still leaves the issue of how to cluster these NIL values to create new entities.

The problem of clustering equivalent entities has been widely studied, and is closely related to the problems of cross-document entity resolution in the NLP community (see e.g., [2, 30]) and record linkage in the DB community (see e.g., [8]). Most methods use some variant of the following basic technique: define (or learn) a pairwise similarity function for comparing two candidate entities, and then apply hierarchical agglomerative clustering (HAC) to find the mention clusters. For example, this approach was used in the Resolver system of [32] to cluster the entities derived from TextRunner's Open IE triples. They defined the similarity of two entities in terms of their string similarity, as well as the similarity of their attribute values. We use a similar technique in our paper; see Section 3 for the details.

We compare our technique to Concept Resolver [18], a state-of-the-art system that clusters entity mentions on NELL data. Concept Resolver operates in two phases. The first phase performs disambiguation under the *one-sense-per-category* assumption. This assumption states that within a given NELL category, noun phrases are unambiguous. For example, *Apple* can be a company and a fruit, but there cannot be two companies called Apple (nor two fruits). We can infer the type of a particular noun phrase by the type signature of the accompanying verb. For instance, for the triple ⟨*Apple, hasCeo, Tim Cook*⟩, the domain of *hasCeo* tells us that *Apple* refers to the company. This triple is then rewritten as ⟨*Apple:company, hasCeo, Tim Cook:person*⟩. The second phase clusters all these type-augmented mentions. For this, they use HAC with a machine learned similarity metric, similar to our approach.

2.3 Clustering relations

There has been less work on clustering synonymous relations than on clustering synonymous entities. The database community has studied schema alignment, but this usually relies on knowing the type signature of the relations, which are unavailable for Open IE triples.

The Resolver system [32] used HAC to cluster Open IE relations in TextRunner data. They used the set of subjects and objects associated with each relation to define a feature vector; they then constructed a generative model (known as the "Extracted Shared Property" model) to compute the probability that two relations are equivalent, based on counting the number of entity pairs that they had in common. Finally they used this similarity metric inside HAC.

The disadvantage of this approach is that it defines the feature vector for a relation in terms of the raw string names for the entities, and these can be very ambiguous. For example, suppose the dataset contains the triples ⟨*Indian Airlines, is headquartered in, Mumbai*⟩ and ⟨*Indian Airlines, has headquarters in, Bombay*⟩. We cannot determine that *is headquartered in* is equivalent to *has headquarters in* unless we know that *Mumbai* is equivalent to *Bombay*.

One solution to this is to jointly cluster the entities and relations at the same time; this has been called "knowledge graph identification" [25]. In this paper, we adopt a simpler two-stage approach. We first perform entity linkage on the triples, mapping the subject and object to a unique id (e.g., both *Bombay* and *Mumbai* map to the Freebase id */m/04vmp*). We then pass these partially-disambiguated triples to the AMIE rule mining system [13, 12], which can discover equivalences between synonymous relations. We then use these learned equivalences to create clusters of semantically equivalent relations. See Section 4 for the details.

The PATTY system [23] uses pattern mining techniques to find subsumption rules between syntactic patterns (e.g., *daughter of* $\sqsubset$ *child of*), extracted from a corpus. Like our approach, PATTY links the arguments of phrases to a KB (YAGO) to find subsumption rules. However, their goal is to construct a taxonomy of verbal phrases, whereas we are interested in finding equivalences between verbal phrases.

The WEBRE approach [22] can cluster verb phrases with close meaning in the presence of ambiguity. For instance, the verb phrase *be part of* holds different semantics in the sentences "New York is part of the United States" (*location* is part of *location*) and "Sun Microsystems is part of Oracle" (*company* is part of *company*). WEBRE first disambiguates the relational concepts, producing a set of typed relations called type A relations (e.g. *company* is part of *company*). Then, WEBRE performs synonym resolution on such concepts. For this purpose, WEBRE uses both the Open IE triples and the source text corpus to construct a hypernym graph, an entity similarity graph and verb phrase similarity graph. Such data structures are used to construct features for a clustering implementation based on HAC. Our approach also deals with ambiguous verbal phrases by enforcing type constraints on the arguments of the equivalence

mappings mined from the Open IE KB. However, unlike WEBRE, our methods rely solely on the Open IE triples.

3. CANONICALIZING NOUN PHRASES

3.1 Mentions

Given an Open IE KB, our first goal is to canonicalize its noun phrases. For simplicity, we concentrate here on canonicalizing the subjects; the same approach can be used to canonicalize the objects. We note that the same string can have different meanings if it appears on two different pages. For example, ⟨*Obama, won, an award*⟩ can refer to Barack Obama or Michelle Obama, depending on where the triple was found. We assume, however, that the same subject phrase on the same Web page will always refer to the same entity. For example, a news article that uses *Obama* to refer to the president may not use that word (without other qualification) to refer to his wife. This is a common assumption in linguistics [14].

With this in mind, we define a mention as a triple $m = (n, u, \mathcal{A})$ where n is a subject noun phrase such as *Barack Obama*, u is the url of a Web document such as *bbc.com* where the mention was found, and $\mathcal{A}$ is the set of attributes about n that were extracted from u. Each attribute is a predicate-object pair such as ⟨*was born in, Honolulu*⟩, or ⟨*won, an award*⟩. Thus, a mention defines the profile of a noun phrase n in a particular Web source u.

3.2 Clustering

Our goal is to partition the set of mentions, so that all mentions in one partition refer to the same real-world entity. This task can be seen as a clustering problem, where the real number of clusters, i.e., the number of different entities in the data, is unknown. To solve this problem, we use Hierarchical Agglomerative Clustering (HAC) on the set of mentions built from the Open IE triples.

In its general formulation, HAC has $O(N^3)$ time complexity, where N is the number of mentions. This makes it inadequate for large datasets. To alleviate this fact, we used token blocking [24], a method that resembles the canopies method introduced in [21]. This method first assigns each mention to one or several groups, called *canopies*. One standard approach is to assign the noun phrases to canopies based on the words that the noun phrases contain. For example, a noun phrase *President Obama* will be assigned to the canopy for *President* and to the canopy for *Obama*. Then, standard HAC is applied within each canopy. This partitions each canopy into a set of clusters. Finally, the clusters that live in different canopies but share a mention are merged. In the example, the cluster {*Barack Obama*, *President Obama*} from the *Obama* canopy will be merged with the cluster {*President Obama*, *US President*} from the *President* canopy. This technique reduces the number of pairwise comparisons required by the standard HAC implementation. Furthermore, it facilitates parallelism, because each canopy can run HAC on a smaller subset of data.

Canopies can drastically affect the recall of the clustering algorithm: If two mentions refer to the same real-world entity, but are assigned to different canopies, then they might never end up in the same cluster. For example, *Mumbai* and *Bombay* will go to different canopies, and unless there is a mention that connects them, they will remain in different clusters. Thus, we face a trade-off, where assigning a mention to a small number of canopies will improve efficiency (decrease the runtime), but hurt recall and precision.

We propose the following solution to this problem, which we call "object canopy expansion": We assign two mentions $m_1 = \langle n_1, w_1, \mathcal{A}_1 \rangle$ and $m_2 = \langle n_2, w_2, \mathcal{A}_2 \rangle$ to the same canopy, if (1) n_1 and n_2 share a word that is not a stopword, or if (2) there exist two objects $o_1 \in \pi_{object}(\mathcal{A}_1), o_2 \in \pi_{object}(\mathcal{A}_2)$ that share a word. In this way, we can merge *Mumbai* and *Bombay*, if, e.g., they both appear in triples ⟨*Mumbai, is located in, the Republic of India*⟩ and ⟨*Bombay, is a city in, India*⟩.

3.3 Similarity Functions

HAC requires a similarity function on the mentions. In this paper, we study different similarity functions, with the goal of determining which ones work best under which conditions. Many of the functions use the Jaccard coefficient:

$$jaccard(\mathcal{S}, \mathcal{S}') = \frac{|\mathcal{S} \cap \mathcal{S}'|}{|\mathcal{S} \cup \mathcal{S}'|}$$

Given two mentions $m = (n, u, \mathcal{A})$ and $m' = (n', u', \mathcal{A}')$, we define the following similarity functions (called *features*):

Attribute Overlap. The attribute overlap is the Jaccard coefficient of the set of attributes:

$$f_{attr}(m, m') := jaccard(\mathcal{A}, \mathcal{A}')$$

Two attributes $(p, o) \in \mathcal{A}$, $(p', o') \in \mathcal{A}'$ are equal if $p = p'$ and $o = o'$.

String Similarity. We use the Jaro-Winkler similarity between n and n' as a feature

$$f_{strsim}(m, m') := jarowinkler(n, n')$$

String Identity. As a special case of the string similarity (and as a baseline), we consider the string identity:

$$f_{strid}(m, m') := \begin{cases} 1, & \text{if } n = n' \\ 0, & \text{else} \end{cases}$$

IDF Token Overlap. If two noun phrases share a word, they are more likely to be similar, but not if many other mentions share that word. For example, the fact that *Rhine River* and *Ruhr River* share the word *River* is not very significant, because this word also appears in a plethora of other river names. Therefore, we introduce a weighted word overlap, in which a word is given more importance if it appears in fewer mentions. We follow the standard Inverse Document Frequency (IDF) approach:

$$f_{itol}(m, m') = \frac{\sum_{w \in w(n) \cap w(n')} log(1 + df(w))^{-1}}{\sum_{w \in w(n) \cup w(n')} log(1 + df(w))^{-1}}$$

Here, $w(\cdot)$ is the set of words of a string, excluding stop words. $df(w)$ is the frequency of the word in the collection of all words that appear in the subjects and the objects of the OpenIE triples.

Word Overlap. If two Web pages share many words, then this can be an indication that two mentions on these pages refer to the same entity. We define

$$f_{wol}(m, m') = jaccard(t(u), t(u'))$$

where $t(\cdot)$ is the set of the top 100 words on a page, ranked in terms of TF-IDF [20].

Entity Overlap. Since words can be ambiguous, we also experimented with an overlap of entities. We applied a standard entity linkage algorithm [11] on the pages, and identified the set $e(u)$ of linked Freebase entities on the page u. Then we define

$$f_{eol}(m, m') = jaccard(e(u), e(u'))$$

Type Overlap. Shared types can be a signal for merging two mentions. We used the results of the type induction algorithm [27] on the verbal phrases of the attributes (as provided by the authors of [27]), and define

$$f_{tol}(m, m') = jaccard(types(\pi_{pred}(\mathcal{A})), types(\pi_{pred}(\mathcal{A}'))$$

So far, we have described how to compute the similarity between two mentions. The similarity between two clusters is calculated using the single linkage criterion [16], that is, the maximum of the intercluster pairwise similarities. Our experience suggests that the complete linkage criterion (the policy that uses the minimum intercluster similarity) and even the average linkage criteria are too conservative and therefore lead to low clustering recall.

3.4 Combined Feature

In addition to the individual features, we also study a combined feature, which is a logistic function:

$$f_{ml}(m, m') = \frac{1}{1 + e^{-f_{sim}(m,m')}}$$

Here, $f_{sim}(m, m')$ is a linear combination of features:

$$f_{sim}(m, m') = c_0 + \sum_{i=1}^{N} c_i f_i(m, m')$$

The $f_1, \ldots, f_N$ are the similarity functions discussed in Section 3.3. Since the word overlap, the entity overlap, and the type overlap require a lot of preprocessing, we also study a simple combined feature that does not make use of these advanced measures, called f_{sml}. In both cases, the weights c_i are determined by training a logistic regression classifier.

To train the combined similarity function, we need labeled training data, i.e., a set of mention pairs that are labeled as being *equal* or *unequal*. Such training data can be obtained, for instance, from approaches that perform entity disambiguation (Section 2.2), i.e., that map Open IE triple subjects to entities in a KB. In our case (i.e., in the case of ReVerb triples), half of the triples already have their subjects mapped to Freebase entities [11]. We use these mappings for training, so that we can partition the remaining, yet-unmapped, mentions into synonymous clusters.

To learn robust weights for our features, we have to make sure that our training set contains hard cases, where the same entity appears with different names, because otherwise we will learn to put too much weight on just the string similarity. To do this, we randomly sample 200 Freebase entities that appear with different names in our triples. For example, our set of entities contains the Greek poet Homer, because he appears as *Homer* and as *Homerus* in our triples. Every one of these names, in turn, can refer to several entities (homonyms). For example, *Homer* does not just refer to the Greek poet, but also to Homer Simpson, the cartoon character from "The Simpsons". Hence, we add to our set also Homer Simpson, and all names that refer to Homer Simpson. This results in 436 entities with 714 names in total. Next we collect all triples that contain one of these 714 names (in mapped form), and construct their mentions (using the provenance information in the Reverb triples). This results in 43K mentions. From these mentions, we can construct a training set of pairs of mentions that are labeled as *equal* or *unequal*. We construct a set of 1137 pairs, balanced between equal and unequal pairs. We also make sure that each entity contributes with at least two examples, because standard random sampling would penalize entities with few mentions. This set of pairs is then used to train our weighted similarity function. In Section 5.1, we compare this learned similarity function to the baseline approaches.

3.5 Canonicalization

Given a cluster of synonym mentions $m = (n, w, \mathcal{A})$, the canonicalization consists of selecting a representative noun phrase $\hat{n}$ that will replace the other noun phrases in the canonicalized KB. We propose a simple approach that selects the noun phrase $\hat{n}$ with the highest number of different Web sources u. In case of a tie, an alternative is to select the longest noun phrase.

4. CANONICALIZING VERBAL PHRASES

In this section, we describe our approach for clustering verbal phrases that have equivalent meaning. The basic idea is to learn rules that predict when one phrase is semantically equivalent to another, and then to perform clustering to enforce transitivity of this relation.

4.1 A semi-canonicalized KB

Our approach to verbal phrase clustering requires that the subjects and objects of the Open IE triples are already clustered. There are two ways to achieve this. Either we can use our noun phrase clustering algorithm of Section 3, or we can make use of the existing mappings to Freebase. We consider both approaches. In particular, for the latter case, we take a subset of all the Reverb triples where the subject was already mapped to Freebase (as provided in the data in [11]), and where we can unambiguously map the object to Freebase using a simple string matching technique.

With either of these techniques, we obtain a "semi-canonicalized" KB, where the subjects and objects are fully canonicalized, and the predicates are still uncanonicalized verbal phrases (the "dual" of NELL). This KB may contain, e.g., ⟨*Barack Obama, was born in, Honolulu*⟩ and ⟨*Barack Obama, 's birthplace is, Honolulu*⟩.

4.2 Rule Mining

Suppose we have the two Open IE relations r=*was born in* and r' =*'s birthplace is*. We would like to discover that these are equivalent, i.e., that $r \sqsubset r'$ and $r' \sqsubset r$, where $r \sqsubset r'$ means that r' subsumes (is more general than) r, i.e., $\forall x, y : r(x, y) \Rightarrow r'(x, y)$. Unfortunately not all triples with r will necessarily appear with r'. Conversely, relations that are very sparse may occur with the same subject and object by chance, even though they are not equivalent. For example, if we find ⟨*Woody Allen, married, Soon-Yi Previn*⟩ and ⟨*Woody Allen, 's stepdaughter, Soon-Yi Previn*⟩, we should not deduce *'s stepdaughter* $\sqsubset$ *married*, even if all triples with the former verbal phrase appear with the latter. Therefore, we resort to a soft approach for subsumption detection that is based on statistical rule mining.

We decided to use the AMIE algorithm [13, 12], which can learn Horn rules such as the following:

$$marriedTo(x, z) \wedge livesIn(z, y) \Rightarrow livesIn(x, y)$$

The left-hand-side of such a rule is called the *body*, and we abbreviate it by $\vec{B}$. All variables are implicitly universally quantified, so that Horn rules are a subset of first order logic statements. A relation subsumption $r \sqsubset r'$ can be expressed as a Horn rule of the form $r(x, y) \Rightarrow r'(x, y)$.

Learning such rules is called inductive logic programming. This requires positive and negative examples. One could take all facts that are absent in the KB as negative examples. However, this violates the Open World Assumption made by KBs. Instead, AMIE makes a more refined assumption called the Partial Completeness Assumption (PCA), which assumes that if we know any facts of relation r for a given entity x, then we know all facts of this relation for x, so missing edges are false, but if we do not know any facts of type r for entity x, then we do not assume missing edges are false. (In [7], PCA is called the "local closed world assumption".) For instance if a KB knows the nationality of a person and a rule predicts a different nationality, this is assumed as a negative example by the PCA. However, if no nationality is known for the person, PCA does not use this absence of information as counter-evidence.

For AMIE, the support of a rule is the number of positive examples covered by the rule:

$$supp(\vec{B} \Rightarrow r(x, y)) := \#(x, y) : \exists z_1, ..., z_m : \vec{B} \wedge r(x, y)$$

Here, $z_1, ..., z_m$ are the free variables of the body, and $\#(x, y) : \mathcal{A}$ is the number of pairs (x, y) that fulfill $\mathcal{A}$. Based on the PCA, AMIE defines the following confidence measure:

$$pcaconf(\vec{B} \Rightarrow r(x, y)) := \frac{supp(\vec{B} \Rightarrow r(x, y))}{\#(x, y') : \exists z_1, ..., z_m, y' : \vec{B} \wedge r(x, y')}$$

The PCA confidence normalizes the support of the rule over the positive examples and the facts assumed as false according to the PCA, thus taking into account that KBs can be incomplete. Although the PCA does not hold for all relations, [13] shows that it leads to high precision and recall for rule mining.

We apply AMIE to our semi-canonicalized KB to mine subsumption rules of the form $r(x, y) \Rightarrow r'(x, y)$. As suggested in [12], we infer equivalence rules of the form $r(x, y) \Leftrightarrow r'(x, y)$ by merging subsumptions in both directions. We score each subsumption with the PCA confidence, and score an equivalence with the minimum of the two subsumption scores. The output of the AMIE system is a set of equivalent verb phrases like *be-named-after(x, y) ⇔ take-its-name-from(x, y)*

4.3 Phrase Clustering

The rule mining has given us a set of weighted equivalence relations between verbal phrases. Since the equivalence relation is transitive, we iteratively merge equivalence mappings with at least one verbal phrase in common. For instance, given the equivalences

stand-for(x, y) ⇔ be-an-acronym-for(x, y)
be-short-for(x, y) ⇔ be-an-acronym-for(x, y)
refer-to(x, y) ⇔ be-short-for(x, y)

we merge the relations *stand-for*, *be-an-acronym-for*, *short-for* and *refer-to* into a single cluster.

4.4 Canonicalization

We propose to canonicalize clusters of verbal phrases by mapping them to Freebase relations. To achieve this, we resort to the ROSA approach [12]. First, we restrict our set of triples to those that have a subject and an object linked to Freebase. Then we join this set with Freebase, so that we obtain a set with facts of the form $vp(x, y)$ and $fr(x, y)$. Here, x and y are Freebase entities, vp is a verbal phrase and fr is a Freebase relation. Then, we run AMIE on our coalesced KB in order to mine cross-ontology equivalence rules of the form $vp(x, y) \Leftrightarrow fr(x, y)$. The rule $bought(y, x) \Leftrightarrow acquiring\_company(x, y)$ is an example. For each cluster of Reverb verb phrases, we collected all the Freebase relations implied by the verbal phrases in these equivalences. In Section 5.2, we show that in some cases it is possible to map clusters unambiguously to Freebase. If a cluster cannot be mapped to Freebase (e.g., because it represents a novel relation that is not part of the Freebase schema), a representative for the cluster can be chosen by selecting either the verb phrase that appears in most equivalences, or the phrase with the largest number of triples.

5. EXPERIMENTS

We conducted two groups of experiments, one to evaluate different entity clustering features, and one to evaluate the relation clustering.

5.1 Entity clustering

5.1.1 Evaluation metrics

To evaluate the quality of a clustering of mentions, we assume each mention m can be mapped to a corresponding entity $e(m)$ from Freebase according to a gold standard clustering E. Each cluster $e \in E$ contains all mentions that map to the same entity. Given this, we can measure precision and recall of the clustering in 3 different ways, which we call macro analysis, micro analysis and pairwise analysis.

We will explain these metrics using the example in Figure 1, which illustrates a set of $|M| = 7$ mentions distributed across $|C| = 3$ clusters. There are $|E| = 3$ Freebase entities, all called "Homer": the Greek poet, the cartoon character Homer Simpson, and the author Homer Hickam. (Each entity also has other aliases.)

Macro-analysis. We define the macro precision of the clustering as the fraction of clusters that are pure, i.e., where all the mentions in the cluster are linked to the same entity:

$$precision_{macro}(C, E) = \frac{|c \in C : \exists_{=1}\, e \in E : e \supseteq c|}{|C|}$$

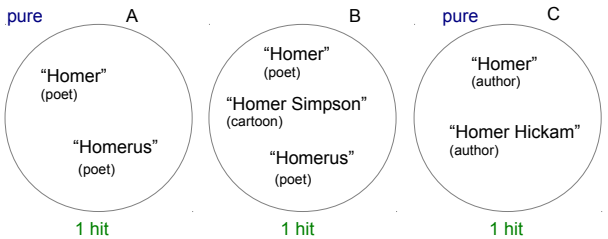

Figure 1: An example of a clustering of N=7 mentions and a gold standard with 3 Freebase entities, labeled in parentheses below the mentions.

Macro recall is calculated by swapping the roles of the ground truth and the resulting clustering, i.e., $recall_{macro}(C, E) = precision_{macro}(E, C)$. This corresponds to the fraction of Freebase entities that get assigned to a unique cluster. In Figure 1, we can see that clusters A and C are pure as they do not mix mentions of different entities. Hence $precision_{macro} = {}^2/_3$. Conversely, the cartoon character and the author are pure entities because they occur only in one cluster, therefore $recall_{macro} = {}^2/_3$.

Micro analysis. Micro precision is defined as the purity [20] of the resulting clustering. This is computed by assuming that the most frequent Freebase entity of the mentions in a cluster is the correct entity. That is, we compute

$$precision_{micro}(C, E) = \frac{1}{N} \sum_{c \in C} max_{e \in E} |c \cap e|$$

where N is the number of mentions in the input. Macro recall is symmetrically defined as $recall_{micro}(C, E) = precision_{micro}(E, C)$. For example, in Figure 1, the most frequent entity for clusters A and B is the poet (2 mentions in each cluster) and for C is the author (2 mentions), so the micro precision is ${}^6/_7$. Analogously $recall_{micro} = {}^5/_7$ because the highest frequencies in a cluster for the entities are: 2 times for the poet, 2 times for the author, and 1 time for the cartoon character.

Pairwise analysis. In the pairwise evaluation, we measure the precision and recall of individual pairwise merging decisions. To be more precise, let us say that two mentions from the same cluster produce a *hit* if they refer to the same Freebase entity. We define the pairwise precision as

$$precision_{pairwise}(C, E) = \frac{\sum_{c \in C} \#hits_c}{\sum_{c \in C} \#pairs_c}$$

$\#pairs_c = |c| \times (|c| - 1)/2$ is the total number of mention pairs in a cluster. Likewise, we define recall as

$$recall_{pairwise}(C, E) = \frac{\sum_{c \in C} \#hits_c}{\sum_{e \in E} \#pairs_e}$$

In Figure 1, clusters A and C produce 1 hit out of 1 pairwise decision, whereas cluster B produces 1 hit out of 3 pairwise decisions. Hence the pairwise precision is $\frac{3}{5}$. To compute $recall_{pairwise}$, we calculate the total number of pairwise decisions in the gold standard E: $\#pairs_{poet} + \#pairs_{author} + \#pairs_{cartoon} = 6 + 1 + 0$, so $recall_{pairwise} = {}^3/_7$.

In all cases, the F1 measure is defined as the harmonic mean of precision and recall.

5.1.2 Clustering ReVerb

ReVerb[1] [10] is a state-of-the-art Open IE system that was run on the Clueweb09 corpus[2]. It produced 3M triples. Half of these triples have their subject linked to Freebase, as in [11]. To evaluate the different clustering features, we built a gold standard as follows: We sampled 150 Freebase entities that appear with at least 2 names in our dataset, and collected all their mentions in our triples. This resulted in 8.5K mentions. We call this dataset *Base*. We then enrich this dataset with all the mentions of homonym entities, as in Section 3.4. We name this dataset *Ambiguous*. This results in 446 Freebase entities and 34K mentions. For both datasets, we constructed a gold standard clustering by grouping those mentions that are linked to the same Freebase entity.

Results on *Base*. We ran our implementation of HAC on the 8.5K mentions (9.1K extractions) in the *Base* dataset. On a Intel Core i7 (8 logical cores, 2.40 GHz) with 16 GB of RAM, our implementation (using the full ML similarity function, plus expanded canopies) created 157 clusters in 54.3 seconds (averaged across 3 runs). Our memory footprint reaches a peek of 5.7GB.

Next, we assess the quality of the different similarity features introduced in Section 3.2. We show the results in Table 1, using a confidence threshold that was chosen to maximize the F1 score. Our first observation is that all the features deliver very good precision. This means that they rarely produce a cluster than contains two different entities. Thus, the features behave rather conservatively.

Let us now look at recall. The macro-recall is very low in general, because as soon as one entity appears in more than one cluster, the metrics decreases (even if all other mentions of the entity are clustered correctly). Let us therefore look at the micro-recall and the pairwise-recall. All features perform decently. This is mainly because all features can build on the pre-clustering that the canopies already established, i.e., every feature is applied only to pairs of mentions with overlapping words. When comparing the recall of the features, the attribute overlap stands out with lowest recall. This is because our triples are very sparse: It rarely happens that two different mentions of the same entity share many attributes. The type overlap, in contrast, works very well: The fact that two mentions with similar names share a type is a strong signal that they refer to the same entity. The word overlap performs well for a similar reason. Among the features that do not require preprocessing, the IDF token overlap is a clear winner.

Rather surprisingly, the combined features (Full ML and Simple ML) are not the best performing methods. We thought that this would be because of the canopies, which make the resulting data distribution at test time quite different from what was used to train the model. To assess this conjecture, we also ran HAC without canopies on this dataset (Table 2). We can observe that the string identity, the IDF tokens overlap, and the attributes overlap are insensitive to the canopies policy. The other features, in contrast, perform much worse without canopies. This suggests that they provide little or no extra evidence of synonymy. They are noisy signals that mainly mislead the ML methods. This, in turn, explains the performance of the ML methods with and without canopies.

Table 3 lists some examples of pure entity clusters that the Simple ML feature could find. We can for instance cluster the mentions "Phoenix Arizona" and "Phoenix" using as signals the tokens overlap and common attributes such as ⟨*located in, Arizona*⟩. Moreover we can avoid mixing these mentions with the mythological creature of the same name. On the other hand, the sparseness of the attribute overlap still leads to losses in recall. For example, we could not cluster the mentions "Beijing National Stadium" and "National Stadium" together, even though they are the same entity.

Results on *Ambiguous*. The *Ambiguous* dataset consists of 34K mentions and 37K triples. With the same setup as for *Base*, our implementation produces 823 clusters in 15.045 minutes on average with a peek memory footprint of 6.5GB. The results are shown in Table 4. Not surprisingly, preci-

[1] `http://OpenIE.cs.washington.edu/`
[2] `http://www.lemurproject.org/clueweb09.php/`

	Macro			Micro			Pairwise		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
String identity	**1.000**	0.436	0.607	**1.000**	0.798	0.888	**1.000**	0.740	0.851
String similarity	0.995	0.658	0.792	0.998	0.844	0.914	0.999	0.768	**0.986**
IDF token overlap	0.994	0.879	0.933	0.996	0.969	0.982	0.999	**0.973**	**0.986**
Attribute overlap	**1.000**	0.05	0.102	**1.000**	0.232	0.377	**1.000**	0.094	0.173
Entity overlap	0.996	0.436	0.607	0.995	0.934	0.964	0.999	0.932	0.964
Type overlap	0.987	**0.926**	**0.956**	0.995	**0.973**	**0.984**	0.999	0.972	0.985
Word overlap	0.988	0.913	0.949	0.995	**0.973**	**0.984**	0.999	**0.973**	**0.986**
Simple ML	0.994	0.899	0.944	0.996	0.972	**0.984**	0.999	**0.973**	**0.986**
Full ML	0.994	0.906	0.948	**1.000**	0.937	0.967	**1.000**	**0.973**	0.869

Table 1: Precision and recall on ReVerb's *Base* dataset. The highest values in each column are in bold.

	Macro			Micro			Pairwise		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
String identity	**1.000**	0.436	0.607	**1.000**	0.798	0.888	**1.000**	0.740	0.851
String similarity	0.948	0.477	0.634	0.971	0.811	0.884	0.973	0.743	0.842
IDF token overlap	0.994	0.879	**0.933**	0.996	0.969	**0.982**	0.999	0.973	**0.986**
Attribute overlap	0.994	0.054	0.102	0.990	0.232	0.376	0.990	0.094	0.172
Entity overlap	0.000	0.805	0.000	0.169	0.987	0.289	0.051	0.981	0.097
Type overlap	0.750	0.980	0.850	0.157	**1.000**	0.272	0.051	0.999	0.097
Word overlap	0.000	**1.000**	0.000	0.157	**1.000**	0.271	0.051	**1.000**	0.097
Simple ML	0.979	0.490	0.653	0.824	0.916	0.868	0.405	0.937	0.565
Full ML	0.990	0.154	0.267	0.776	0.889	0.829	0.396	0.931	0.555

Table 2: Precision and recall on ReVerb's *Base* dataset, without canopies. Highest values in bold.

sion is lower on this dataset. This is mainly caused by the single linkage criterion for clusters. Under this policy, the similarity of a pair of clusters is determined by the highest intercluster similarity score. While this aggressive strategy was shown to improve recall significantly, it also propagates errors more easily. Furthermore, this phenomenon is amplified by the reclustering phase and the ambiguity of the test set. To see this, consider a set of mentions with labels *Barack Obama*, *Michelle Obama*, and *Obama*, the latter referring ambiguously to both Barack and Michelle Obama. A single clustering error on the ambiguous mentions (*Obama*) will move the three entities in the same cluster, and thus decrease precision. Conversely, the baseline approach is less sensitive to ambiguity in this particular case because the precision will only penalize the merge of the ambiguous mention *Obama*, as *Barack Obama* and *Michelle Obama* will be never clustered together. Hence, all features produce lower precision. The attribute overlap has highest precision – but this is mainly because it is a very sparse feature that hesitates to merge mentions. Let us therefore look at the micro-F1 and the pairwise-F1. We see that the IDF token overlap is the strongest feature, even in presence of ambiguity. It is followed by the combined features. For comparison, the table also shows the Simple ML without the "object canopy expansion" (Section 3.2). We see that the expansion increases recall slightly, and has a negligible effect on precision. Therefore, we conclude that the technique is indeed useful. Table 3 shows some examples of impure clusters.

Pure clusters
{*Phoenix Arizona*, *Phoenix*}
{*Hannibal Hamlin*, *Hamlin*, *Hannibal*}
{*The Colorado Rockies*, *The Rockies*}
Impure clusters
{*John's Gospel*, *John Peel*, *Peel*}
{*Suns*, *Phoenix Coyotes*, *Phoenix Suns*}
{*Ethanol*, *Cellulosic Ethanol* }

Table 3: Some examples of successfull and unsuccessful synonym identification

Lessons Learned. Our study of features shows that, among the more advanced features, the type overlap and the word overlap produce significant leverage if combined with canopies – both on the random dataset and on the ambiguous one. They are closely followed by the IDF token overlap, which stands out as the strongest simple feature, with and without canopies. The combined features also perform decently. All in all, the best features can cluster entities in the general case with nearly 100% precision, and a pairwise-recall of more than 98%.

5.1.3 Clustering NELL

The NELL project [5] also extracts triples from the ClueWeb09 corpus. The Concept Resolver approach [18] aims to cluster the entities of these triples. Concept Resolver operates under the one-sense-per-category assumption, which states that within one category, names are unambiguous. Once the type for a noun phrase has been extracted, Concept Resolver collects all the type compatible triples about the noun phrase and builds a "sense" for that NP, the equivalent of what we call a "mention". The authors of [18] provided us with the data and the training examples used to evaluate Concept Resolver. As described in Section 3.4, we used the triples to construct mentions.

The challenge is to cluster together several names that refer to the same entity. Since not all the triples contained the source from which they were extracted, we defined a default source for those without provenance. We also restricted our dataset to the categories considered in the evaluation of Concept Resolver. The resulting dataset consists of 18K triples and 20K mentions.

	Macro			Micro			Pairwise		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
String identity	0.734	0.390	0.510	0.932	0.771	0.844	0.942	0.565	0.706
String similarity	0.607	0.442	0.511	0.792	0.873	0.831	0.809	0.574	0.671
IDF token overlap	0.643	0.509	0.568	0.913	0.847	**0.879**	0.900	0.703	**0.789**
Attribute overlap	**0.997**	0.083	0.153	**0.998**	0.162	0.279	**0.997**	0.024	0.047
Entity overlap	0.905	0.480	**0.627**	0.663	0.939	0.777	0.458	0.892	0.606
Type overlap	0.467	0.917	0.619	0.626	**0.970**	0.760	0.401	0.914	0.558
Word overlap	0.390	**0.926**	0.549	0.625	**0.970**	0.760	0.401	0.915	0.557
Simple ML, no obj.can.	0.711	0.444	0.546	0.808	0.909	0.855	0.630	0.889	0.738
Simple ML	0.709	0.444	0.546	0.808	0.923	0.862	0.649	0.968	0.777
Full ML	0.685	0.552	0.611	0.671	0.955	0.788	0.302	**0.989**	0.463

Table 4: Precision and recall on ReVerb's *Ambiguous* dataset. The highest values in each column are in bold.

	Micro-evaluation			Pairwise		
	Precision	Recall	F1	Precision	Recall	F1
Simple ML	0.660	0.578	0.616	0.376	0.188	0.250
Concept Resolver	0.778	0.633	0.699	0.542	0.335	0.415
IDF Token Overlap	0.700	0.475	0.566	0.356	0.067	0.113

Table 5: Comparison of entity clustering methods on the NELL data.

Gold Standard. Concept Resolver comes with a manually compiled set of noun phrases that should be clustered together as one entity. We estimated our precision by sampling a random subset of 100 clusters from the output and checking them manually. For Concept Resolver we report the precision value from [18], averaged across all categories and weighted by the number of senses per category.

We note that our notion of precision is not fully comparable to theirs: the one-sense-per-category assumption merges all mentions of the same type into one mention – no matter if the triples come from different sources. This can cause problems. For example, *Obama:Person* is assumed to map to only one entity, whereas we allow it to map to multiple entities, depending on which page the mention was seen. Moreover, Concept Resolver removes singleton clusters from the evaluation of precision. For us, in contrast, such a singleton cluster can correspond to several successfully merged mentions. Therefore, we restricted our manual evaluation to non-singleton clusters according to Concept Resolver and used the IDF token overlap as baseline. Concept Resolver does not report values for the macro dimension, we show results for the micro and pairwise evaluation.

Results. We ran our synonym resolution machinery on the NELL triples and computed precision and recall. The results are shown in Table 5. The Simple ML feature can achieve decent precision and recall, albeit inferior to the values of Concept Resolver. We believe this is because the logistic regression model implemented by Concept Resolver leverages information from the ontology, which is not possible in an Open IE scenario. Whereas our Simple ML uses a Jaccard score on the attributes of mentions as signal for synonymy, Concept Resolver also builds features from the properties of the relations. For instance, their approach will penalize the similarity for the mentions *Auburn* and *Auburn Hills* as they have different values for the functional predicate *cityLocatedInState* (the first located in Maine and the second in Michigan) even if they have other attributes in common (e.g. both located in USA). Additionally, Concept Resolver makes use of inverse and quasi-inverse functions as they can identify mentions uniquely. The fact that the mentions *Mumbai* and *Bombay* share the inverse functional predicate ⟨*cityCapitalofState, Maharastra*⟩ is used as strong evidence for synonymy. Still, our Simple ML and the IDF token overlap deliver reasonable results even without this additional information.

Lessons Learned. We can see here that the relation schema that Concept Solver uses improves the entity clustering results. If such data is not available, a synonym resolution approach based on Open IE attributes and string similarity features can deliver reasonable results. Even a simple baseline such as the IDF token overlap should not be outright discarded for the task of synonym resolution.

5.2 Relation clustering

5.2.1 Dataset

Since the relations of NELL are already canonicalized, we run the relation clustering only on the ReVerb set of triples. As discussed in Section 4.1, the relation clustering requires a semi-canonicalized KB, in which the subjects and objects have been canonicalized. There are two ways to achieve this: either by our entity clustering ("Clustered KB") or by a mapping to Freebase ("Linked KB"). The Clustered KB was constructed by sampling 25K freebase ids from the Linked KB and gathering all their triples. This results in 600K triples. Both the Clustered and the Linked KB are given as input to the AMIE system in order to mine relation equivalences. We ran AMIE using a support threshold of 5, meaning that two verbal phrases must have at least 5 pairs in common. This also implies that relations with less than 5 cases are discarded. The Linked KB has 33215 of such relations, the Clustered KB has 17259.

5.2.2 Ambiguous phrases

[22] suggests that approximately 22% of the Reverb phrases are polysemous. The phrase *belongs-to*, e.g., conveys different meanings in the sentences "The Wii belongs to Nintendo" (*invention created by organization*) and "Mallorca belongs to Spain" (*island belongs to country*). Polysemy hurts precision, since phrases with unrelated meanings

	Conf.	Phrases	Clusters	Precision			In Freebase	Triples covered
				Macro	Micro	Pairwise		
Linked KB	0.8	522	118	0.900	0.936	0.946	18%	15%
	0.5	967	143	0.896	0.690	0.337	25%	29%
Linked KB (types)	0.8	752	303	0.946	0.980	0.997	9%	21%
	0.5	1185	319	0.861	0.892	0.779	14%	27%
Clustered KB	0.8	826	234	0.940	0.716	0.273	6%	16%
	0.5	1185	264	0.813	0.665	0.292	8%	33%

Table 6: Quality of relation clusters for two different confidence thresholds.

Verb phrases	Freebase relation
be an abbreviation-for, be known as, stand for, be an acronym for	-
be spoken in, be the official language of, be the national language of	`location.country.official_language`
be bought, acquire	`organization.organization.acquired_by`

Table 7: Examples of clusters of verbal phrases. The last two were mapped to Freebase.

will be transitively clustered as synonyms due to the polysemic relations. To alleviate this effect, we also run AMIE on triples from Linked KB where the entities are augmented with types. This option makes AMIE enforce type constraints on the arguments of the equivalence mappings when possible. Thus, a single verbal phrase can generate multiple specific relations that differ only in their signatures. Since the Freebase ontology has approximately 23K different data types, we allowed type enhacement only with the most common types, i.e., person, organization, location, and string.

5.2.3 Evaluation metrics

Since evaluating recall would require identifying all the relations in the set of 1.3M triples, we focus on measures of precision. As before, we can measure the precision of a relation cluster at the macro, micro or pairwise level. The pairwise precision measures the quality of a set of clusters as the ratio of correct pairwise merges. A pairwise merge is counted as correct if the corresponding verbal phrases mean the same, or if one of them is slightly more general than the other. For instance, the phrases *be-spoken-in* and *be-the-official-language-of* count as a correct merge. The micro-precision assumes the most frequent meaning in a cluster as ground truth. Hence, it is calculated by adding up the frequency of the most common meaning in each of the clusters and dividing this number by the total number of phrases. Conversely, the macro-precision is the ratio of pure clusters, i.e., the proportion of clusters where all the phrases belong to the same meaning. For micro and macro precision, phrases were labeled with the most general meaning.

5.2.4 Results

On the Clustered KB, AMIE mined 3.5K equivalence mappings, whereas mining the Linked KB produced 4.3K equivalence rules. When the type enhancement is enabled, the number rises to 22K mappings. For example, we find *use-truck-for* $\Leftrightarrow$ *use-van-for* with confidence 1.0 and support 19, or *stand-for* $\Leftrightarrow$ *be-an-acronym-for* with confidence 0.88 and support 44. With the type enhancement, AMIE can discriminate between a country changing location, e.g., "Israel moved to Mont Hor", a person visiting a place, e.g., "Barack Obama moved to Los Angeles", and an organization changing location, e.g., "Fostoria Glass moved to Moundsville" . This results in different equivalence rules for the phrase *move-to* such as *moved-to(location* $\rightarrow$ *location)* $\Leftrightarrow$ *located-in(location* $\rightarrow$ *location)* (support 8, confidence 0.5) and *moved-to(person* $\rightarrow$ *location)* $\Leftrightarrow$ *move-permanently-to(person* $\rightarrow$ *location)* (support 5, confidence 0.5). In these examples our coarse-grained type constraints are enough to avoid mixing phrases that denote permanent location (e.g. *situated-in*) with phrases that denote place of residence (e.g. *now-lives-in*).

The mappings have different confidence scores, and therefore we tested the phrase clustering at two confidence thresholds: 0.8 and 0.5. Table 6 shows the results, together with the number of clusters and phrases. As we see, the precision of our clusters is very good, meaning that we do not merge phrases that do not belong together. Naturally, a higher confidence threshold always leads to fewer phrases being clustered, and to fewer clusters. Our results are better on the cleaner Linked KB than on the Clustered KB. We can also observe the benefit of the type enhancement. In general, we can cluster only a very small portion of the verbal phrases. However, our clusters are non-trivial and contain an average of 4-5 phrases.

Comparison to WEBRE. We compare our results to the WEBRE system [22]. We use the precision on the Linked KB with types because the type-enhanced phrases resemble the type A relations introduced by WEBRE. To be comparable, we report a weighted micro-precision, where the correct assignments of phrases to clusters are weighted by the number of triples with the verbal phrase. We get a score of 0.981, which is slightly better than WEBRE's score of 0.897. Nevertheless, this comparison must be taken with a grain of salt because the evaluation performed in WEBRE is somewhat different from ours (see Section 5 in [22]). First, their method to generate typed verbal phrases is different. Second, we could not have access to their gold standard for precision and recall. Third, the micro-precision formula of WEBRE uses are more granular definition of synonymy: a phrase can be a synonym of the true relation of a cluster (score 1), somehow related (score 0.5) or unrelated (score 0). Nevertheless, it is safe to say that our approach is not far off from the state-of-the-art in terms of precision.

Mapping to Freebase. As described in Section 4.4, we used AMIE and ROSA rules [12] to find equivalences be-

tween verbal phrases and Freebase relations. We ran AMIE on a combination of Freebase and Reverb with support threshold 5, producing 5.1K cross-ontology mappings. We then applied the same confidence thresholds as for relation clustering (0.5 and 0.8) and used the rules to map the clusters of verb phrases to Freebase. We counted clusters that were mapped to one Freebase relation or to two mutually inverse Freebase relations as "correctly mapped". For example, Freebase expresses the fact that Steven Spielberg directed the Titanic movie with a pair of mutually inverse relations, ⟨*Steven Spielberg, directed, Titanic*⟩ and ⟨*Titanic, directed by, Steven Spielberg*⟩. The last column in Table 6 shows the proportion of triples whose relation could be mapped. We find a large number of very interesting mappings, some of which are shown in Table 7. Going manually through the mappings, we find an average precision of 88% for the Clustered KB with threshold 0.8.

Lessons Learned. We conclude that rule mining can be used to cluster verbal phrases, and to map them to canonical Freebase relations. A cleaner KB and the type enhancement both help. This method can produce such clusterings and mappings only for a small portion of the verbal phrases, but it can do so at a high precision and for a significant percentage of the Reverb triples.

6. CONCLUSIONS

We have shown that it is possible, using fairly simple and standard machine learning techniques, to identify synonym mentions in a reasonable fraction of the triples coming from standard Open IE systems, such as Reverb and NELL. Our results suggest that, even with a certain level of ambiguity, the IDF token overlap is the strongest signal of synonymy for noun phrases on the Web, whereas more sophisticated features extracted from the sources are insufficient for this task on their own. We also provided useful and novel insights about the impact of canopies in the performance of Hierarchical Agglomerative Clustering, a standard technique for record linkage and identification of synonyms. The resulting clusters of entities and relations are semantically meaningful; some of them correspond to existing entities and predicates in Freebase, but others are novel extensions. We believe this hybrid approach — whereby we use high recall extractors, followed by clustering methods to improve the precision — shows great promise for bridging the gap between Open and Closed IE methods for knowledge base construction.

7. 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 *ISWC*, 2007.

[2] A. Bagga and B. Baldwin. Entity-based cross-document coreferencing using the vector space model. In *COLING*, 1998.

[3] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *IJCAI*, 2007.

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

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

[6] L. Del Corro and R. Gemulla. Clausie: clause-based open information extraction. In *WWW*, 2013.

[7] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*, 2014.

[8] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD*, 2005.

[9] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam. Open Information Extraction: the Second Generation. In *IJCAI*, 2011.

[10] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *EMNLP*, 2011.

[11] E. Gabrilovich, M. Ringgaard, and A. Subramanya. FACC1: Freebase annotation of ClueWeb corpora, version 1, June 2013.

[12] L. A. Galárraga, N. Preda, and F. M. Suchanek. Mining rules to align knowledge bases. In *AKBC*, 2013.

[13] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek. AMIE: Association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*, 2013.

[14] W. A. Gale, K. W. Church, and D. Yarowsky. One sense per discourse. In *Workshop on Speech and Natural Language*, 1992.

[15] B. Hachey, W. Radford, J. Nothman, M. Honnibal, and J. Curran. Evaluating entity linking with wikipedia. *Artificial Intelligence*, 194, 2013.

[16] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[17] J. Hoffart, Y. Altun, and G. Weikum. Discovering emerging entities with ambiguous names. In *WWW*, 2014.

[18] J. Krishnamurthy and T. M. Mitchell. Which noun phrases denote which concepts? In *HLT*, 2011.

[19] T. Lin, Mausam, and O. Etzioni. Entity linking at web scale. In *AKBC-WEKEX*, 2012.

[20] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[21] A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD*, 2000.

[22] B. Min, S. Shi, R. Grishman, and C. Y. Lin. Ensemble semantics for large-scale unsupervised relation extraction. In *EMNLP-CoNLL*, 2012.

[23] N. Nakashole, G. Weikum, and F. Suchanek. Patty: A taxonomy of relational patterns with semantic types. In *EMNLP*, 2012.

[24] G. Papadakis, E. Ioannou, C. Niederée, and P. Fankhauser. Efficient entity resolution for large heterogeneous information spaces. In *WSDM*, 2011.

[25] J. Pujara, H. Miao, L. Getoor, and W. Cohen. Knowledge graph identification. In *ISWC*, 2013.

[26] L. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation to wikipedia. In *NAACL*, 2011.

[27] A. Ritter and O. Etzioni. A latent dirichlet allocation method for selectional preferences. In *ACL*, 2010.

[28] M. Schmitz, R. Bart, S. Soderland, O. Etzioni, et al. Open language learning for information extraction. In *EMNLP-CoNLL*, 2012.

[29] F. Suchanek, G. Kasneci, and G. Weikum. YAGO - A Core of Semantic Knowledge. In *WWW*, 2007.

[30] M. Wick, S. Singh, and A. McCallum. A discriminative hierarchical model for fast coreference at large scale. In *ACL*, 2012.

[31] F. Wu and D. S. Weld. Open information extraction using wikipedia. In *ACL*, 2010.

[32] A. Yates and O. Etzioni. Unsupervised methods for determining object and relation synonyms on the web. *J. Artif. Int. Res.*, 34(1), Mar. 2009.