Ensemble Semantics for Large-scale Unsupervised Relation Extraction

Bonan Min[1*] **Shuming Shi**[2] **Ralph Grishman**[1] **Chin-Yew Lin**[2]

[1]New York University
New York, NY, USA
{min,grishman}@cs.nyu.edu

[2]Microsoft Research Asia
Beijing, China
{shumings,cyl}@microsoft.com

Abstract

Discovering significant types of relations from the web is challenging because of its open nature. Unsupervised algorithms are developed to extract relations from a corpus without knowing the relations in advance, but most of them rely on tagging arguments of predefined types. Recently, a new algorithm was proposed to jointly extract relations and their argument semantic classes, taking a set of relation instances extracted by an open IE algorithm as input. However, it cannot handle polysemy of relation phrases and fails to group many similar ("synonymous") relation instances because of the sparseness of features. In this paper, we present a novel unsupervised algorithm that provides a more general treatment of the polysemy and synonymy problems. The algorithm incorporates various knowledge sources which we will show to be very effective for unsupervised extraction. Moreover, it explicitly disambiguates polysemous relation phrases and groups synonymous ones. While maintaining approximately the same precision, the algorithm achieves significant improvement on recall compared to the previous method. It is also very efficient. Experiments on a real-world dataset show that it can handle 14.7 million relation instances and extract a very large set of relations from the web.

1 Introduction

Relation extraction aims at discovering semantic relations between entities. It is an important task that has many applications in answering factoid questions, building knowledge bases and improving search engine relevance. The web has become a massive potential source of such relations. However, its open nature brings an open-ended set of relation types. To extract these relations, a system should not assume a fixed set of relation types, nor rely on a fixed set of relation argument types.

The past decade has seen some promising solutions, unsupervised relation extraction (URE) algorithms that extract relations from a corpus without knowing the relations in advance. However, most algorithms (Hasegawa et al., 2004, Shinyama and Sekine, 2006, Chen et. al, 2005) rely on tagging predefined types of entities as relation arguments, and thus are not well-suited for the open domain.

Recently, Kok and Domingos (2008) proposed Semantic Network Extractor (SNE), which generates argument semantic classes and sets of synonymous relation phrases at the same time, thus avoiding the requirement of tagging relation arguments of predefined types. However, SNE has 2 limitations: 1) Following previous URE algorithms, it only uses features from the set of input relation instances for clustering. Empirically we found that it fails to group many relevant relation instances. These features, such as the surface forms of arguments and lexical sequences in between, are very sparse in practice. In contrast, there exist several well-known corpus-level semantic resources that can be automatically derived from a source corpus and are shown to be useful for generating the key elements of a relation: its 2 argument semantic classes and a set of synonymous phrases. For example, semantic classes can be derived from a source corpus with contextual distributional similarity and web table co-occurrences. The "*synonymy*" [1] problem for clustering relation instances

[*] Work done during an internship at Microsoft Research Asia

Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1027–1037, Jeju Island, Korea, 12–14 July 2012.

could potentially be better solved by adding these resources. 2) SNE assumes that each entity or relation phrase belongs to exactly one cluster, thus is not able to effectively handle *polysemy* of relation phrases[2]. An example of a polysemous phrase is *be the currency of* as in 2 triples <*Euro, be the currency of, Germany*> and <*authorship, be the currency of, science*>. As the target corpus expands from mostly news to the open web, polysemy becomes more important as input covers a wider range of domains. In practice, around 22% (section 3) of relation phrases are polysemous. Failure to handle these cases significantly limits its effectiveness.

To move towards a more general treatment of the *polysemy* and *synonymy* problems, we present a novel algorithm WEBRE for open-domain large-scale unsupervised relation extraction without predefined relation or argument types. The contributions are:

- WEBRE incorporates a wide range of corpus-level semantic resources for improving relation extraction. The effectiveness of each knowledge source and their combination are studied and compared. To the best of our knowledge, it is the first to combine and compare them for unsupervised relation extraction.
- WEBRE explicitly disambiguates polysemous relation phrases and groups synonymous phrases, thus fundamentally it avoids the limitation of previous methods.
- Experiments on the Clueweb09 dataset (lemurproject.org/clueweb09.php) show that WEBRE is effective and efficient. We present a large-scale evaluation and show that WEBRE can extract a very large set of high-quality relations. Compared to the closest prior work, WEBRE significantly improves recall while maintaining the same level of precision. WEBRE is efficient. To the best of our knowledge, it handles the largest triple set to date (7-fold larger than largest previous effort). Taking 14.7 million triples as input, a complete run with one CPU core takes about a day.

[1] We use the term *synonymy* broadly as defined in Section 3.

[2] A cluster of relation phrases can, however, act as a whole as the phrase cluster for 2 different relations in SNE. However, this only accounts for 4.8% of the polysemous cases.

2 Related Work

Unsupervised relation extraction (URE) algorithms (Hasegawa et al., 2004; Chen et al., 2005; Shinyama and Sekine, 2006) collect pairs of co-occurring entities as relation instances, extract features for instances and then apply unsupervised clustering techniques to find the major relations of a corpus. These UREs rely on tagging a predefined set of argument types, such as Person, Organization, and Location, in advance. Yao et al. 2011 learns fine-grained argument classes with generative models, but they share the similar requirement of tagging coarse-grained argument types. Most UREs use a quadratic clustering algorithm such as Hierarchical Agglomerate Clustering (Hasegawa et al., 2004, Shinyama and Sekine, 2006), K-Means (Chen et al., 2005), or both (Rosenfeld and Feldman, 2007); thus they are not scalable to very large corpora.

As the target domain shifts to the web, new methods are proposed without requiring predefined entity types. Resolver (Yates and Etzioni, 2007) resolves objects and relation synonyms. Kok and Domingos (2008) proposed Semantic Network Extractor (SNE) to extract concepts and relations. Based on second-order Markov logic, SNE used a bottom-up agglomerative clustering algorithm to jointly cluster relation phrases and argument entities. However, both Resolver and SNE require each entity and relation phrase to belong to exactly one cluster. This limits their ability to handle polysemous relation phrases. Moreover, SNE only uses features in the input set of relation instances for clustering, thus it fails to group many relevant instances. Resolver has the same sparseness problem but it is not affected as much as SNE because of its different goal (synonym resolution).

As the preprocessing instance-detection step for the problem studied in this paper, open IE extracts relation instances (in the form of triples) from the open domain (Etzioni et al., 2004; Banko et al., 2007; Fader et al., 2011; Wang et al. 2011). For efficiency, they only use shallow features. Reverb (Fader et al., 2011) is a state-of-the-art open domain extractor that targets verb-centric relations, which have been shown in Banko and Etzioni (2008) to cover over 70% of open domain relations. Taking their output as input, algorithms have been proposed to resolve objects and relation synonyms (Resolver), extract semantic networks

(SNE), and map extracted relations into an existing ontology (Soderland and Mandhani, 2007).

Recent work shows that it is possible to construct semantic classes and sets of similar phrases automatically with data-driven approaches. For generating semantic classes, previous work applies distributional similarity (Pasca, 2007; Pantel et al., 2009), uses a few linguistic patterns (Pasca 2004; Sarmento et al., 2007), makes use of structure in webpages (Wang and Cohen 2007, 2009), or combines all of them (Shi et al., 2010). Pennacchiotti and Pantel (2009) combines several sources and features. To find similar phrases, there are 2 closely related tasks: paraphrase discovery and recognizing textual entailment. Data-driven paraphrase discovery methods (Lin and Pantel, 2001; Pasca and Dienes, 2005; Wu and Zhou, 2003; Sekine, 2005) extends the idea of distributional similarity to phrases. The Recognizing Textual Entailment algorithms (Berant et al. 2011) can also be used to find related phrases since they find pairs of phrases in which one entails the other.

To efficiently cluster high-dimensional datasets, canopy clustering (McCallum et al., 2000) uses a cheap, approximate distance measure to divide data into smaller subsets, and then cluster each subset using an exact distance measure. It has been applied to reference matching. The second phase of WEBRE applies the similar high-level idea of partition-then-cluster for speeding up relation clustering. We design a graph-based partitioning subroutine that uses various types of evidence, such as shared hypernyms.

3 Problem Analysis

The basic input is a collection of relation instances (triples) of the form <ent_1, *ctx*, ent_2>. For each triple, *ctx* is a relational phrase expressing the relation between the first argument ent_1 and the second argument ent_2. An example triple is <*Obama, win in, NY*>. The triples can be generated by an open IE extractor such as TextRunner or Reverb. Our goal is to automatically build a list of relations $R = \{< \text{ent}_1, ctx, \text{ent}_2 >\} \approx^{3} < C_1, P, C_2 >$ where P is the set of relation phrases, and C_1 and C_2 are two argument classes. Examples of triples and relations *R* (as Type B) are shown in Figure 1.

[3] This approximately equal sign connects 2 possible representations of a relation: as a set of triple instances or a triple with 2 entity classes and a relation phrase class.

The first problem is the *polysemy* of relation phrases, which means that a relation phrase *ctx* can express different relations in different triples. For example, the meaning of *be the currency of* in the following two triples is quite different: <*Euro, be the currency of, Germany*> and <*authorship, be the currency of, science*>. It is more appropriate to assign these 2 triples to 2 relations "*a currency is the currency of a country*" and "*a factor is important in an area*" than to merge them into one. Formally, a relation phrase *ctx* is *polysemous* if there exist 2 different relations $< C_1, P, C_2 >$ and $< C_1^{'}, P^{'}, C_2^{'} >$ where $ctx \in P \cap P^{'}$. In the previous example, *be the currency of* is polysemous because it appears in 2 different relations.

Polysemy of relation phrases is not uncommon. We generate clusters from a large sample of triples with the assistance of a soft clustering algorithm, and found that around 22% of relation phrases can be put into at least 2 disjoint clusters that represent different relations. More importantly, manual inspection reveals that some common phrases are polysemous. For example, *be part of* can be put into a relation "a *city is located in a country"* when connecting *Cities* to *Countries*, and another relation *"a company is a subsidiary of a parent company"* when connecting *Companies* to *Companies.* Failure to handle polysemous relation phrases fundamentally limits the effectiveness of an algorithm. The WEBRE algorithm described later explicitly handles polysemy and synonymy of relation phrases in its first and second phase respectively.

The second problem is the "*synonymy"* of relation instances. We use the term *synonymy* broadly and we say 2 relation instances are synonymous if they express the same semantic relation between the same pair of semantic classes. For example, both <*Euro, be the currency used in, Germany*> and <*Dinar, be legal tender in, Iraq*> express the relation <*Currencies, be currency of, Countries*>. Solving this problem requires grouping synonymous relation phrases and identifying argument semantic classes for the relation.

Various knowledge sources can be derived from the source corpus for this purpose. In this paper we pay special attention to incorporating various semantic resources for relation extraction. We will show that these semantic sources can significantly improve the coverage of extracted relations and the

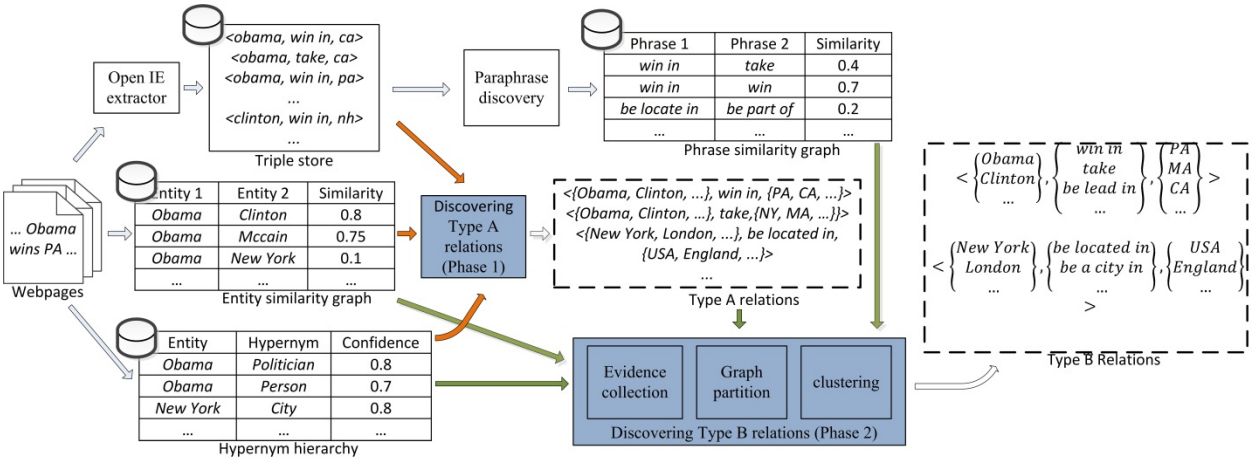

Figure 1. Overview of the WEBRE algorithm (Illustrated with examples sampled from experiment results). The tables and rectangles with a database sign show knowledge sources, shaded rectangles show the 2 phases, and the dotted shapes show the system output, a set of Type A relations and a set of Type B relations. The orange arrows denote resources used in phase 1 and the green arrows show the resources used in phase 2.

best performance is achieved when various resources are combined together.

4 Mining Relations from the Web

We first describe relevant knowledge sources, and then introduce the WEBRE algorithm, followed by a briefly analysis on its computational complexity.

4.1 Knowledge Sources

Entity similarity graph We build two similarity graphs for entities: a distributional similarity (DS) graph and a pattern-similarity (PS) graph. The DS graph is based on the distributional hypothesis (Harris, 1985), saying that terms sharing similar contexts tend to be similar. We use a text window of size 4 as the context of a term, use Pointwise Mutual Information (PMI) to weight context features, and use Jaccard similarity to measure the similarity of term vectors. The PS graph is generated by adopting both sentence lexical patterns and HTML tag patterns (Hearst, 1992; Kozareva et al., 2008; Zhang et al., 2009; Shi et al., 2010). Two terms (*T*) tend to be semantically similar if they co-occur in multiple patterns. One example of sentence lexical patterns is *(such as | including) T{,T}* (and|,|.)*. HTML tag patterns include tables, dropdown boxes, etc. In these two graphs, nodes are entities and the edge weights indicate entity similarity. In all there are about 29.6 million nodes and 1.16 billion edges.

Hypernymy graph Hypernymy relations are very useful for finding semantically similar term pairs. For example, we observed that a small city in UK and another small city in Germany share common hypernyms such as *city*, *location*, and *place*. Therefore the similarity between the two cities is large according to the hypernymy graph, while their similarity in the DS graph and the PS graph may be very small. Following existing work (Hearst, 1992, Pantel & Ravichandran 2004; Snow et al., 2005; Talukdar et al., 2008; Zhang et al., 2011), we adopt a list of lexical patterns to extract hypernyms. The patterns include *NP {,} (such as) {NP,}* {and|or} NP*, *NP (is|are|was|were|being) (a|an|the) NP*, etc. The hypernymy graph is a bipartite graph with two types of nodes: entity nodes and label (hypernym) nodes. There is an edge *(T, L)* with weight *w* if *L* is a hypernym of entity *T* with probability *w*. There are about 8.2 million nodes and 42.4 million edges in the hypernymy graph. In this paper, we use the terms *hypernym* and *label* interchangeably.

Relation phrase similarity: To generate the pairwise similarity graph for relation phrases with regard to the probability of expressing the same relation, we apply a variant of the DIRT algorithm (Lin and Pantel, 2001). Like DIRT, the paraphrase discovery relies on the distributional hypothesis, but there are a few differences: 1) we use stemmed lexical sequences (relation phrases) instead of dependency paths as phrase candidates because of the very large scale of the corpus. 2) We used ordered

pairs of arguments as features of phrases while DIRT uses them as independent features. We empirically tested both feature schemes and found that using ordered pairs results in likely paraphrases but using independent features the result contains general inference rules[4].

4.2 WEBRE for Relation Extraction

WEBRE consists of two phases. In the first phase, a set of semantic classes are discovered and used as argument classes for each relation phrase. This results in a large collection of relations whose arguments are pairs of semantic classes and which have exactly one relation phrase. We call these relations the *Type A* relations. An example Type A relation is *<{New York, London…}, be locate in, {USA, England, …}>*. During this phase, polysemous relation phrases are disambiguated and placed into multiple Type A relations. The second phase is an efficient algorithm which groups similar Type A relations together. This step enriches the argument semantic classes and groups synonymous relation phrases to form relations with multiple expressions, which we called *Type B* relations. Both Type A and Type B relations are system outputs since both are valuable resources for downstream applications such as QA and Web Search. An overview of the algorithm is shown in Figure 1. Here we first briefly describe a clustering subroutine that is used in both phases, and then describe the algorithm in detail.

To handle polysemy of objects (e.g., entities or relations) during the clustering procedure, a key building block is an effective Multi-Membership Clustering algorithm (*MMClustering*). For simplicity and effectiveness, we use a variant of Hierarchical Agglomerative Clustering (HAC), in which we first cluster objects with HAC, and then reassign each object to additional clusters when its similarities with these clusters exceed a certain threshold[5]. In the remainder of this paper, we use *{C} = MMClustering({object}, SimFunc, α)* to represent running *MMClustering* over a set of objects, with threshold *α* to generate a list of clusters *{C}* of the objects, given the pairwise object similarity function *SimFunc*. Our implementation uses HAC with average linkage since empirically it performs well.

Discovering Type A Relations The first phase of the relation extraction algorithm generates Type A relations, which have exactly one relation phrase and two argument entity semantic classes. For each relation phrase, we apply a clustering algorithm on each of its two argument sets to generate argument semantic classes. The *Phase 1* algorithm processes relation phrases one by one. For each relation phrase *ctx*, step 4 clusters the set $\{ent_1\}$ using *MMClustering* to find left-hand-side argument semantic classes $\{C_1\}$. Then for each cluster *C* in $\{C_1\}$, it gathers the right-hand-side arguments which appeared in some triple whose left hand-side-side argument is in *C*, and puts them into $\{ent_2'\}$. Following this, it clusters $\{ent_2'\}$ to find right-hand-side argument semantic classes. This results in pairs of semantic classes which are arguments of *ctx*. Each relation phrase can appear in multiple non-overlapping Type A relations. For example, *<Cities, be part of, Countries>* and *<Companies, be part of, Companies>* are different Type A relations which share the same relation phrase *be part of*. In the pseudo code, *SimEntFunc* is encoded in the entity similarity graphs.

Algorithm ***Phase 1: Discovering Type A relations***

Input: set of triples $T=\{<ent_1, ctx, ent_2>\}$
entity similarity function *SimEntFunc*
Similarity threshold *α*

Output: list of Type A relations $\{<C_1, ctx, C_2>\}$

Steps:

01. For each relation phrase *ctx*
02. $\{ent_1, ctx, ent_2\}$ = set of triples sharing *ctx*
03. $\{ent_1\}$ = set of ent_1 in $\{ent_1, ctx, ent_2\}$
04. $\{C_1\}$ = *MMClustering*($\{ent_1\}$, *SimEntFunc*, *α*)
05. For each *C* in $\{C_1\}$
06. $\{ent_2'\}$ = set of ent_2 $s.t. \exists < ent_1, ctx, ent_2 > \in T \wedge ent_1 \in C_1$
07. $\{C_2\}$ = *MMClustering*($\{ent_2'\}$, *SimEntFunc*, *α*)
08. For each C_2 in $\{C_2\}$
09. Add $<C_1, ctx, C_2>$ into $\{<C_1, ctx, C_2>\}$
10. Return $\{<C_1, ctx, C_2>\}$

Discovering Type B Relations The goal of *phase 2* is to merge similar Type A relations, such as *<Cities, be locate in, Countries>* and *<Cities, be city of, Countries>*, to produce Type B relations, which have a set of synonymous relation phrases and more complete argument entity classes. The challenge for this phase is to cluster a very large

[4] For example, *be part of* has ordered argument pairs *<A, B>* and *<C, D>*, and *be not part of* has ordered argument pairs *<A, D>* and *<B, C>*. If arguments are used as independent features, these two phrases shared the same set of features *{A, B, C, D}*. However, they are inferential (complement relationship) rather than being similar phrases.

[5] This threshold should be slightly greater than the clustering threshold for HAC to avoid generating duplicated clusters.

set of Type A relations, on which it is infeasible to run a clustering algorithm that does pairwise all pair comparison. Therefore, we designed an evidence-based partition-then-cluster algorithm.

The basic idea is to heuristically partition the large set of Type A relations into small subsets, and run clustering algorithms on each subset. It is based on the observation that most pairs of Type A relations are not similar because of the sparseness in the entity class and the relation semantic space. If there is little or no evidence showing that two Type A relations are similar, they can be put into different partitions. Once partitioned, the clustering algorithm only has to be run on each much smaller subset, thus computation complexity is reduced.

The 2 types of evidence we used are shared members and shared hypernyms of relation arguments. For example, 2 Type A relations r_1=<*Cities, be city of, Countries*> and r_2=<*Cities, be locate in, Countries*> share a pair of arguments <*Tokyo, Japan*>, and a pair of hypernyms <"*city*", "*country*">. These pieces of evidence give us hints that they are likely to be similar. As shown in the pseudo code, shared arguments and hypernyms are used as independent evidence to reduce sparseness.

Algorithm ***Phase 2: Discovering Type B relations***	
Input:	Set of Type A relations *{r}={<C_1, ctx, C_2>}* Relation similarity function *SimRelationFunc* Map from entities to their hypernyms: $M_{entity2label}$ Similarity threshold α Edge weight threshold μ
Variables	*G(V, E)* = weighted graph in which *V={r}*
Output:	Set of Type B relations *{<C_1, P, C_2>}*
Steps:	
01.	*{<ent, {r'}>}* = build inverted index from argument *ent* to set of Type A relations *{r'}* on *{<C_1, ctx, C_2>}*
02	*{<l, {r'}>}* = build inverted index from hypernym *l* of arguments to set of Type A relations *{r'}* on *{<C_1, ctx, C_2>}* with map $M_{entity2label}$
03.	For each *ent* in *{<ent, {r'}>}*
04.	For each pair of r_1 and r_2 s.t. $r_1 \in \{r'\} \wedge r_2 \in \{r'\}$
05.	*weight_edge(<r_1, r_2>) += weight (ent)*
06.	For each *l* in *{<l, {r'}>}*
07.	For each pair of r_1 and r_2 s.t. $r_1 \in \{r'\} \wedge r_2 \in \{r'\}$
08.	*weight_edge(<r_1, r_2>) += weight (l)*
09.	For each edge <r_1, r_2> in *G*
10.	If *weight_edge(<r_1, r_2>) < μ*
11.	Remove edge <r_1, r_2> from *G*
12.	*{CC}= DFS(G)*
13.	For each connected component *CC* in *{CC}*
14.	*{<C_1, ctx, C_2>}* = vertices in *CC*
15.	*{<C_1', P', C_2'>} = MMClustering({<C_1, ctx, C_2>}, SimRelationFunc, α)*
16.	Add *{<C_1', P', C_2'>}* into *{<C_1, P, C_2>}*
17.	Return *{<C_1, P, C_2>}*

Steps 1 and 2 build an inverted index from evidence to sets of Type A relations. On the graph *G* whose vertices are Type A relations, steps 3 to 8 set the value of edge weights based on the strength of evidence that shows the end-points are related. The weight of evidence *E* is calculated as follows:

$$weight(E) = \frac{\#\ shared\ tuples\ in\ which\ E\ appears\ in}{\max(\#\ classes\ E\ appears\ in)}$$

The idea behind this weighting scheme is similar to that of TF-IDF in that the weight of evidence is higher if it appears more frequently and is less ambiguous (appeared in fewer semantic classes during clustering of phase 1). The weighting scheme is applied to both shared arguments and labels.

After collecting evidence, we prune (steps 9 to 11) the edges with a weight less than a threshold μ to remove noise. Then a Depth-First Search (DFS) is called on *G* to find all Connected Components *CC* of the graph. These *CC*s are the partitions of likely-similar Type A relations. We run *MMClustering* on each *CC* in *{CC}* and generate Type B relations (step 13 to step 16). The similarity of 2 relations (*SimRelationFunc*) is defined as follows:

$$sim(< C_1, P, C_2 >, < C_1', P', C_2' >) = \begin{cases} 0, & if\ sim(P, P') < \sigma \\ \min\big(sim(C_1, C_1'), sim(C_2, C_2')\big), & else \end{cases}$$

4.3 Computational Complexity

WEBRE is very efficient since both phases decompose the large-clustering task into much smaller clustering tasks over partitions. Given *n* objects for clustering, a hierarchical agglomerative clustering algorithm requires $O(n^2)$ pairwise comparisons. Assuming the clustering task is split into subtasks of size $n_1, n_2, \ldots, n_k$, thus the computational complexity is reduced to $O(\sum_1^k n_i^2)$. Ideally each subtask has an equal size of n/k, so the computational complexity is reduced to $O(n^2/k)$, a factor of k speed up. In practice, the sizes of partitions are not equal. Taking the partition sizes observed in the experiment with 0.2 million Type A relations as input, the phase 2 algorithm achieves around a 100-fold reduction in pairwise comparisons compared to the agglomerative clustering algorithm. The combination of phase 1 and phase 2 achieves more than a 1000-fold reduction in pairwise comparison, compared to running an agglomerative clustering algorithm directly on 14.7 million triples. This reduction of computational

complexity makes the unsupervised extraction of relations on a large dataset a reality. In the experiments with 14.7 million triples as input, phase 1 finished in 22 hours, and the phase 2 algorithm finished in 4 hours with one CPU core.

Furthermore, both phases can be run in parallel in a distributed computing environment because data is partitioned. Therefore it is scalable and efficient for clustering a very large number of relation instances from a large-scale corpus like the web.

5 Experiment

Data preparation We tested WEBRE on resources extracted from the English subset of the Clueweb09 Dataset, which contains 503 million webpages. For building knowledge resources, all webpages are cleaned and then POS tagged and chunked with in-house tools. We implemented the algorithms described in section 4.1 to generate the knowledge sources, including a hypernym graph, two entity similarity graphs and a relation phrase similarity graph.

We used Reverb Clueweb09 Extractions 1.1 (downloaded from *reverb.cs.washington.edu*) as the triple store (relation instances). It is the complete extraction of Reverb over Clueweb09 after filtering low confidence and low frequency triples. It contains 14.7 million distinct triples with 3.3 million entities and 1.3 million relation phrases. We choose it because 1) it is extracted by a state-of-the-art open IE extractor from the open-domain, and 2) to the best of our knowledge, it contains the largest number of distinct triples extracted from the open-domain and which is publicly available.

Evaluation setup The evaluations are organized as follows: we evaluate Type A relation extraction and Type B relation extraction separately, and then we compare WEBRE to its closest prior work SNE. Since both phases are essentially clustering algorithms, we compare the output clusters with human labeled gold standards and report performance measures, following most previous work such as Kok and Domingos (2008) and Hasegawa et al. (2004). Three gold standards are created for evaluating Type A relations, Type B relations and the comparison to SNE, respectively. In the experiments, we set $\alpha=0.6$, $\mu=0.1$ and $\sigma=0.02$ based on trial runs on a small development set of 10k relation instances. We filtered out the Type A relations and Type B relations which only contain 1 or 2 triples since most of these relations are not different from a single relation instance and are not very interesting. Overall, 0.2 million Type A relations and 84,000 Type B relations are extracted.

Evaluating Type A relations To understand the effectiveness of knowledge sources, we run *Phase 1* multiple times taking entity similarity graphs (matrices) constructed with resources listed below:

- **TS**: Distributional similarity based on the triple store. For each triple <ent_1, *ctx*, ent_2>, features of ent_1 are *{ctx}* and *{ctx* ent_2*}*; features of ent_2 are *{ctx}* and *{*ent_1 *ctx}*. Features are weighted with PMI. Cosine is used as similarity measure.
- **LABEL:** The similarity between two entities is computed according to the percentage of top hypernyms they share.
- **SIM:** The similarity between two entities is the linear combination of their similarity scores in the distributional similarity graph and in the pattern similarity graph.
- **SIM+LABEL** SIM and LABEL are combined. Observing that SIM generates high quality but overly fine-grained semantic classes, we modify the entity clustering procedure to cluster argument entities based on SIM first, and then further clustering the results based on LABEL.

The outputs of these runs are pooled and mixed for labeling. We randomly sampled 60 relation phrases. For each phrase, we select the 5 most frequent Type A relations from each run ($4\times5=20$[6] Type A relations in all). For each relation phrase, we ask a human labeler to label the mixed pool of Type A relations that share the phrase: 1) The labelers[7] are asked to first determine the major semantic relation of each Type A relation, and then label the triples as *good*, *fair* or *bad* based on whether they express the major relation. 2) The labeler also reads all Type A relations and manually merges the ones that express the same relation. These 2 steps are repeated for each phrase. After labeling, we create a gold standard *GS1*, which contains roughly 10,000 triples for 60 relation phrases. On average, close to 200 triples are manu-

[6] Here 4 means the 4 methods (the bullet items above) of computing similarity.

[7] 4 human labelers perform the task. A portion of the judgments were independently dual annotated; inter-annotator agreement is 79%. Moreover, each judgment is cross-checked by at least one more annotator, further improving quality.

ally labeled and clustered for each phrase. This creates a large data set for evaluation.

We report micro-average of precision, recall and F1 on the 60 relation phrases for each method. Precision (P) and Recall (R) of a given relation phrase is defined as follows. Here R_A and R'_A represents a Type A relation in the algorithm output and *GS1*, respectively. We use *t* for triples and *s(t)* to represent the score of the labeled triple *t*. *s(t) is* set to *1.0, 0.5* or *0* for *t* labeled as good, fair and bad, respectively.

$$P = \frac{\sum_{R_A} \sum_{t \in R_A} s(t)}{\sum_{R_A} |R_A|}, R = \frac{\sum_{R_A} \sum_{t \in R_A} s(t)}{\sum_{R'_A} \sum_{t' \in R'_A} s(t')}$$

The results are in table 1. Overall, LABEL performs 53% better than TS in F-measure, and SIM+LABEL performs the best, 8% better than LABEL. Applying a simple sign test shows both differences are clearly significant ($p<0.001$). Surprisingly, SIM, which uses the similarity matrix extracted from full text, has a F1 of 0.277, which is lower than TS. We also tried combining TS and LABEL but did not find encouraging performance compared to SIM+LABEL.

Algorithm	Precision	Recall	F1
TS	0.842 (0.886)	0.266	0.388
LABEL	0.855 (0.870)	0.481	0.596
SIM	0.755 (**0.964**)	0.178	0.277
SIM+LABEL	0.843 (0.872)	**0.540**	**0.643**

Table 1. Phase 1 performance (averaged on multiple runs) of the 4 methods. The highest performance numbers are in bold. (The number in parenthesis is the micro-average when empty-result relation phrases are not considered for the method).

Among the 4 methods, SIM has the highest precision (0.964) when relation phrases for which it fails to generate any Type A relations are excluded, but its recall is low. Manual checking shows that SIM tends to generate overly fine-grained argument classes. If fine-grained argument classes or extremely high-precision Type A relations are preferred, SIM is a good choice. LABEL performs significantly better than TS, which shows that hypernymy information is very useful for finding argument semantic classes. However, it has coverage problems in that the hypernym finding algorithm failed to find any hypernym from the corpus for some entities. Following up, we found that SIM+LABEL has similar precision and the highest recall. This shows that the combination of semantic spaces is very helpful. The significant recall improvement from TS to SIM+LABEL shows that the corpus-based knowledge resources significantly reduce the data sparseness, compared to using features extracted from the triple store only. The result of the phase 1 algorithm with SIM+LABEL is used as input for phase 2.

Evaluating Type B relations The goal is 2-fold: 1) to evaluate the phase 2 algorithm. This involves comparing system output to a gold standard constructed by hand, and reporting performance; 2) to evaluate the quality of Type B relations. For this, we will also report triple-level precision.

We construct a gold standard *GS2*[8] for evaluating Type B relations as follows: We randomly sampled 178 Type B relations, which contain 1547 Type A relations and more than 100,000 triples. Since the number of triples is very large, it is infeasible for labelers to manually cluster triples to construct a gold standard. To report precision, we asked the labelers to label each Type A relation contained in this Type B relation as good, fair or bad based on whether it expresses the same relation. For recall evaluation, we need to know how many Type A relations are missing from each Type B relation. We provide the full data set of Type A relations along with three additional resources: 1) a tool which, given a Type A relation, returns a ranked list of similar Type A relations based on the pairwise relation similarity metric in section 4, 2) DIRT paraphrase collection, 3) WordNet (Fellbaum, 1998) synsets. The labelers are asked to find similar phrases by checking phrases which contain synonyms of the tokens in the query phrase. Given a Type B relation, ideally we expect the labelers to find all missing Type A relations using these resources. We report precision (P) and recall (R) as follows. Here R_B and R'_B represent Type B relations in the algorithm output and *GS2*, respectively. R_A and R'_A represent Type A relations. $s(R_A)$ denotes the score of R_A. It is set to 1.0, 0.5 and 0 for good, fair or bad respectively.

$$P = \frac{\sum_{R_B} \sum_{R_A \in R_B} |R_A| \cdot s(R_A)}{\sum_{R_B} \sum_{R_A \in R_B} |R_A|}, R = \frac{\sum_{R_B} \sum_{R_A \in R_B} |R_A| \cdot s(R_A)}{\sum_{R'_B} \sum_{R'_A \in R'_B} |R'_A|}$$

We also ask the labeler to label at most 50 randomly sampled triples from each Type B relation, and calculate triple-level precision as the ratio of the sum of scores of triples over the number of

[8] 3 human labelers performed the task. A portion of the judgments were independently dual annotated; inter-annotator agreement is 73%. Similar to labeling Type A relations, each judgment is cross-checked by at least one more annotator, further improving quality.

Argument 1	Relation phrase	Argument 2
marijuana, caffeine, nicotine…	*result in, be risk factor for, be major cause of…*	*insomnia, emphysema, breast cancer,…*
C# 2.0, php5, java, c++, …	*allow the use of, also use, introduce the concept of…*	*destructors, interfaces, template,…*
clinton, obama, mccain, …	*win, win in, take, be lead in,…*	*ca, dc, fl, nh, pa, va, ga, il, nc,…*

Table 3. Sample Type B relations extracted.

sampled triples. We use P_{ins} to represent the precision calculated based on labeled triples. Moreover, as we are interested in how many phrases are found by our algorithm, we also include R_{phrase}, which is the recall of synonymous phrases. Results are shown in Table 2.

Interval	*P*	*R* (R_{phrase})	*F1*	P_{ins}	count
[3, 5)	0.913	0.426 (0.026)	0.581	0.872	52149
[5, 10)	0.834	0.514 (0.074)	0.636	0.863	21981
[10, 20)	0.854	0.569 (0.066)	0.683	0.883	6277
[20, 50)	0.899	0.675 (0.406)	0.771	0.894	2630
[50, +∞)	0.922	0.825 (0.594)	0.871	0.929	1089
Overall	0.897	0.684 (0.324)	0.776	0.898	84126

Table 2. Performance for Type B relation extraction. The first column shows the range of the maximum sizes of Type A relations in the Type B relation. The last column shows the number of Type B relations that are in this range. The number in parenthesis in the third column is the recall of phrases.

The result shows that WEBRE can extract Type B relations at high precision (both *P* and P_{ins}). The overall recall is 0.684. Table 2 also shows a trend that if the maximum number of Type A relation in the target Type B relation is larger, the recall is better. This shows that the recall of Type B relations depends on the amount of data available for that relation. Some examples of Type B relations extracted are shown in Table 3.

Comparison with SNE We compare WEBRE's extracted Type B relations to the relations extracted by its closest prior work SNE[9]. We found SNE is not able to handle the 14.7 million triples in a foreseeable amount of time, so we randomly sampled 1 million (1M) triples[10] and test both algorithms on this set. We also filtered out result clusters which have only 1 or 2 triples from both system outputs. For comparison purposes, we constructed a gold standard *GS3* as follows: randomly select 30 clusters from both system outputs, and then find similar clusters from the other system output, followed by manually refining the clusters by merging similar ones and splitting non-coherent clusters. *GS3* contains 742 triples and 135 clusters. We report triple-level pairwise precision, recall and F1 for both algorithms against *GS3*, and report results in Table 4. We fine-tuned SNE (using grid search, internal cross-validation, and coarse-to-fine parameter tuning), and report its best performance.

Algorithm	Precision	Recall	F1
WEBRE	0.848	0.734	0.787
SNE	0.850	0.080	0.146

Table 4. Pairwise precision/recall/F1 of WEBRE and SNE.

Table 4 shows that WEBRE outperforms SNE significantly in pairwise recall while having similar precision. There are two reasons. First, WEBRE makes use of several corpus-level semantic sources extracted from the corpus for clustering entities and phrases while SNE uses only features in the triple store. These semantic resources significantly reduced data sparseness. Examination of the output shows that SNE is unable to group many triples from the same generally-recognized fine-grained relations. For example, SNE placed relation instances *<Barbara, grow up in, Santa Fe>* and *<John, be raised mostly in, Santa Barbara>* into 2 different clusters because the arguments and phrases do not share features nor could be grouped by SNE's mutual clustering. In contrast, WEBRE groups them together. Second, SNE assumes a relation phrase to be in exactly one cluster. For example, SNE placed *be part of* in the phrase cluster *be city of* and failed to place it in another cluster *be subsidiary of*. This limits SNE's ability to placing relation instances with polysemous phrases into correct relation clusters.

It should be emphasized that we use pairwise precision and recall in table 4 to be consistent with the original SNE paper. Pairwise metrics are much more sensitive than instance-level metrics, and penalize recall exponentially in the worst case[11] if an algorithm incorrectly splits a coherent cluster; therefore the absolute pairwise recall difference

[9] Obtained from alchemy.cs.washington.edu/papers/kok08

[10] We found that SNE's runtime on 1M triples varies from several hours to over a week, depending on the parameters. The best performance is achieved with runtime of approximately 3 days. We also tried SNE with 2M triples, on which many runs take several days and show no sign of convergence. For fairness, the comparison was done on 1M triples.

[11] Pairwise precision and recall are calculated on all pairs that are in the same cluster, thus are very sensitive. For example, if an algorithm incorrectly split a cluster of size *N* to a smaller main cluster of size *N/2* and some constant-size clusters, pairwise recall could drop to as much as ¼ of its original value.

should not be interpreted as the same as the instance-level recall reported in previous experiments. On 1 million triples, WEBRE generates 12179 triple clusters with an average size[12] of 13 while SNE generate 53270 clusters with an average size 5.1. In consequence, pairwise recall drops significantly. Nonetheless, at above 80% pairwise precision, it demonstrates that WEBRE can group more related triples by adding rich semantics harvested from the web and employing a more general treatment of polysemous relation phrases. On 1M triples, WEBRE finished in 40 minutes, while the run time of SNE varies from 3 hours to a few days.

6 Conclusion

We present a fully unsupervised algorithm WEBRE for large-scale open-domain relation extraction. WEBRE explicitly handles polysemy relations and achieves a significant improvement on recall by incorporating rich corpus-based semantic resources. Experiments on a large data set show that it can extract a very large set of high-quality relations.

Acknowledgements

Supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory (AFRL) contract number FA8650-10-C-7058. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL, or the U.S. Government.

References

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open Information Extraction from the Web. In Proceedings of IJCAI 2007.

Michele Banko and Oren Etzioni. 2008. The Tradeoffs Between Open and Traditional Relation Extraction. In Proceedings of ACL 2008.

Jonathan Berant, Ido Dagan and Jacob Goldberger. 2011. Global Learning of Typed Entailment Rules. In Proceedings of ACL 2011.

Razvan Bunescu and Raymond J. Mooney. 2004. Collective Information Extraction with Relational Markov Networks. In Proceedings of ACL 2004.

Jinxiu Chen, Donghong Ji, Chew Lim Tan, Zhengyu Niu. 2005. Unsupervised Feature Selection for Relation Extraction. In Proceedings of IJCNLP 2005.

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-scale information extraction in KnowItAll (preliminary results). In Proceedings of WWW 2004.

Oren Etzioni, Michael Cafarella, Doug Downey, AnaMaria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld and Alexander Yates. 2005. Unsupervised named-entity extraction from the Web: An Experimental Study. In Artificial Intelligence, 165(1):91-134.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying Relations for Open Information Extraction. In Proceedings of EMNLP 2011.

Christiane Fellbaum (Ed.). 1998. WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

Zelig S. Harris. 1985. Distributional Structure. The Philosophy of Linguistics. New York: Oxford University Press.

Takaaki Hasegawa, Satoshi Sekine, Ralph Grishman . 2004.Discovering Relations among Named Entities from Large Corpora. In Proceedings of ACL 2004.

Marti A. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In Proceedings of COLING 1992.

Stanley Kok and Pedro Domingos. 2008. Extracting Semantic Networks from Text via Relational Clustering. In Proceedings of ECML 2008.

Zornitsa Kozareva, Ellen Riloff, Eduard Hovy. 2008. Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs. In Proceedings of ACL 2008.

Dekang Lin and Patrick Pantel. 2001. DIRT – Discovery of Inference Rules from Text. In Proceedings of KDD 2001.

Andrew McCallum, Kamal Nigam and Lyle Ungar. 2000. Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching. In Proceedings of KDD 2000.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu and Vishnu Vyas. 2009. Web-Scale Distributional Similarity and Entity Set Expansion. In Proceedings of EMNLP 2009.

[12] The clusters which have only 1 or 2 triples are removed and not counted here for both algorithms.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In Proceedings of KDD2002.

Patrick Pantel and Deepak Ravichandran. 2004. Automatically Labeling Semantic Classes. In Proceedings of HLT/NAACL-2004.

Marius Pasca. 2004. Acquisition of Categorized Named Entities for Web Search, In Proceedings of CIKM 2004.

Marius Pasca. 2007. Weakly-supervised discovery of named entities using web search queries. In Proceedings of CIKM 2007.

Marius Pasca and Peter Dienes. 2005. Aligning needles in a haystack: Paraphrase acquisition across the Web. In Proceedings of IJCNLP 2005.

Marco Pennacchiotti and Patrick Pantel. 2009. Entity Extraction via Ensemble Semantics. In Proceedings of EMNLP 2009.

Benjamin Rosenfeld and Ronen Feldman. 2007. Clustering for Unsupervised Relation Identification. In Proceedings of CIKM 2007.

Luis Sarmento, Valentin Jijkoun, Maarten de Rijke and Eugenio Oliveira. 2007. "More like these": growing entity classes from seeds. In Proceedings of CIKM 2007.

Satoshi Sekine. 2005. Automatic paraphrase discovery based on context and keywords between NE pairs. In Proceedings of the International Workshop on Paraphrasing, 2005.

Shuming Shi, Huibin Zhang, Xiaojie Yuan, Ji-Rong Wen. 2010. Corpus-based Semantic Class Mining: Distributional vs. Pattern-Based Approaches. In Proceedings of COLING 2010.

Yusuke Shinyama, Satoshi Sekine. 2006. Preemptive Information Extraction using Unrestricted Relation Discovery, In Proceedings of NAACL 2006.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning Syntactic Patterns for Automatic Hypernym Discovery. In Proceedings of In NIPS 17, 2005.

Stephen Soderland and Bhushan Mandhani. 2007. Moving from Textual Relations to Ontologized Relations. In Proceedings of the 2007 AAAI Spring Symposium on Machine Reading.

Partha Pratim Talukdar, Joseph Reisinger, Marius Pasca, Deepak Ravichandran, Rahul Bhagat and Fernando Pereira. 2008. Weakly-Supervised Acquisition of Labeled Class Instances using Graph Random Walks. In Proceedings of EMNLP 2008.

David Vickrey, Oscar Kipersztok and Daphne Koller. 2010. An Active Learning Approach to Finding Related Terms. In Proceedings of ACL 2010.

Vishnu Vyas and Patrick Pantel. 2009. SemiAutomatic Entity Set Refinement. In Proceedings of NAACL/HLT 2009.

Vishnu Vyas, Patrick Pantel and Eric Crestan. 2009, Helping Editors Choose Better Seed Sets for Entity Set Expansion, In Proceedings of CIKM 2009.

Richard C. Wang and William W. Cohen. 2007. Language- Independent Set Expansion of Named Entities Using the Web. In Proceedings of ICDM 2007.

Richard C. Wang and William W. Cohen. 2009. Automatic Set Instance Extraction using the Web. In Proceedings of ACL-IJCNLP 2009.

Wei Wang, Romaric Besançon and Olivier Ferret. 2011. Filtering and Clustering Relations for Unsupervised Information Extraction in Open Domain. In Proceedings of CIKM 2011.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using Wikipedia. In Proceedings of ACL 2010.

Hua Wu and Ming Zhou. 2003. Synonymous collocation extraction using translation information. In Proceedings of the ACL Workshop on Multiword Expressions: Integrating Processing 2003.

Limin Yao, Aria Haghighi, Sebastian Riedel, Andrew McCallum. 2011. Structured Relation Discovery Using Generative Models. In Proceedings of EMNLP 2011.

Alexander Yates and Oren Etzioni. 2007. Unsupervised Resolution of Objects and Relations on the Web. In Proceedings of HLT-NAACL 2007.

Fan Zhang, Shuming Shi, Jing Liu, Shuqi Sun, Chin-Yew Lin. 2011. Nonlinear Evidence Fusion and Propagation for Hyponymy Relation Mining. In Proceedings of ACL 2011.

Huibin Zhang, Mingjie Zhu, Shuming Shi, and Ji-Rong Wen. 2009. Employing Topic Models for Pattern-based Semantic Class Discovery. In Proceedings of ACL 2009.