Mining for Analogous Tuples from an Entity-Relation Graph

Danushka Bollegala Mitsuru Kusumoto Yuichi Yoshida Ken-ichi Kawarabayashi
The University of Tokyo, Hongo, Bunkyo-ku, 7-3-1, Tokyo, 113-8656, Japan.
Kyoto University, 36-1 Yoshida-Honmachi, Sakyo-ku, Kyoto, 606-8501, Japan.
National Institute of Informatics, 2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430, Japan.
Japan Science and Technology Agency, ERATO, Kawarabayashi Large Graph Project.

Abstract

The ability to recognize analogies is an important factor that is closely related to human intelligence. Verbal analogies have been used for evaluating both examinees at university entrance exams as well as algorithms for measuring relational similarity. However, relational similarity measures proposed so far are confined to measuring the similarity between *pairs* of words. Unfortunately, such pairwise approaches ignore the rich relational structure that exists in real-world knowledge bases containing millions of entities and semantic relations. We propose a method to efficiently identify analogous entity tuples from a given entity-relation graph.

First, we present an efficient approach for extracting potential analogous tuples from a given entity-relation graph. Second, to measure the structural similarity between two tuples, we propose two types of kernel functions: *vertex-feature* kernels, and *edge-feature* kernels. Moreover, we combine those kernels to construct composite kernels that simultaneously consider both vertex and edge features. Experimental results show that our proposed method accurately identifies analogous tuples and significantly outperforms a state-of-the-art pairwise relational similarity measure, extended to tuples.

Introduction

Analogies arise when there are one or more similar semantic relations that are shared between two different knowledge bases. Studies on cognitive science have shown that the ability to detect analogies materializes in humans at an early stage [Kotovsky and Gentner, 1996]. Although exact repetitions of past experiences are rare, we tend to compare first-time events we encounter with our past experiences and efficiently find analogous solutions [Gentner and Gentner, 1983]. Detecting analogies has been an important research topic in Artificial Intelligence for many years. Given two knowledge bases, Structure Mapping Theory [Gentner, 1983] states that analogies arise when we can find a mapping between the semantic relations that exist among the entities in one knowledge base (source) and the other (target). For example, consider the *solar system* and the *classical atomic* system. In the solar system, the relation ***X** revolves around **Y*** exists between the *earth* and the *sun*, whereas in the atomic system the same relation exists between an *electron* and the *nucleus*. Although the attributes such as the *mass* and the *size* of the entities involved in the relation ***X** revolves around **Y*** are very different in the two systems compared, we can identify the same semantic relation in the two systems. Consequently, the solar system and the classical atomic system can be considered as analogous. However, structure mapping is an NP-hard problem [Veale and Keane, 2003] for which several sub-optimal heuristic algorithms have been proposed such as the structure mapping engine [Falkenhainer *et al.*, 1989].

The ability to detect analogies has been used as a measure of human intelligence. For example, verbal analogies have been used for evaluating candidates applying for universities in examinations such as the Scholastic Aptitude Test (SAT) and the Graduate Record Examination (GRE) [Turney *et al.*, 2003]. For example, given the word pair (*ostrich,bird*) as the question, examinees must identify the word pair that is analogous to the question from multiple choices. In this example, (*lion,cat*) is a correct answer because the semantic relation ***X** is a large **Y*** exists between *ostrich* and *bird*, as well as between *lion* and *cat*. The two word pairs in this example form a proportional analogy.

Detecting verbal analogies is a challenging task even for humans. For example, the average score (i.e. percentage of correct answers to the total number of questions) obtained by high school students on the verbal analogy questions in SAT is as low as 57% [Turney, 2005]. Despite those challenges, the ability to detect analogies has resulted in novel applications such as the *latent relational search engine* [Duc *et al.*, 2011], an information retrieval system based on verbal analogies. For example, given a query such as (*Apple*, *iPad*), (*Microsoft*, ?), the relational search engine first identifies the relationship between the first two entities (i.e. *Apple* and *iPad*), and then finds an entity that has a similar relation with the third entity (i.e. *Microsoft*). In this example, the tablet device produced by *Apple* is the *iPad*, and the tablet device produced by *Microsoft*, which is *Surface* will be the correct answer. Here, the relationship ***X** produces tablet device **Y*** exists between the first two entities. In social networks, identification of analogous communities might lead to more accurate friend recommendations in future.

Despite those successful applications, there are several fun-

damental limitations in the existing approaches proposed for detecting verbal analogies. They consider only analogies involving two *pairs* of words. However, in real-world we observe analogies that involve more than two entities. For example, consider the two tuples (*Steve Jobs, Apple, Tim Cook*) and (*Bill Gates, Microsoft, Steve Balmer*). Here, in each tuple between the first and second entities, second and the third entities, and first and the third entities we have respectively founderOf relation, ceoOf relation, and successorOf relation. Therefore, the triangular relational structure that exists in one tuple can be directly mapped to the other, indicating a strong analogy between the two tuples. However, if we take a pairwise approach, then we loose the rich analogical structure that exists among the three entities in each tuple. For example, given three entities X, Y, and Z and we are told that X is the CEO of company Y, and Y acquired Z, then the likelihood of X also being the CEO of Z is high. However, if we consider individual pairwise combinations of the three entities, then we loose this vital piece of information.

Prior work on relational similarity assume the word pairs between which we must compute relational similarity to be given. However, in our setting, we are given a large set of entities represented by an entity-relation graph from which we must find analogical entity tuples. Although theoretically we can first generate all possible tuples between a given set of entities, and then search for the most relationally similar ones to discover verbal analogies, this approach does not scale-up for large sets of entities. For example, the number of possible pairs between n items grows in the order of $\mathcal{O}(n^2)$, whereas the problem is further aggravated when we consider tuples that contain three or more entities.

We propose a method that overcomes the above-mentioned limitations and can discover a large number of analogical tuples from an entity-relation graph. Our contributions can be summarized as follows:

- First, given an entity-relation graph, in which vertices correspond to entities and edges correspond to lexical-syntactic patterns that represent numerous semantic relations between entities, we propose a set of efficient heuristics to identify potential analogous tuples.

- Second, we propose numerous kernel functions to measure the degree of analogy between two tuples. For this purpose, we present two types of kernel functions: *Vertex-Feature Kernels* and *Edge-Feature Kernels*. Moreover, we combine those two types of kernels to construct *Composite Kernels*.

We describe and evaluate the proposed method for tuples that consists of three entities. Although the proposed method can be directly extended to find analogical tuples consisting more than three entities, we leave this to future work. Experiments conducted using a large entity-relation graph that contains $749,364$ vertices and $3,862,331$ edges, show that the proposed method significantly outperforms competitive baselines and Latent Relational Analysis (LRA) [Turney, 2006] – a state-of-the-art pairwise relational similarity measure extended to measure the degree of analogy between two tuples.

Related Work

The current state-of-the-art method for measuring the relational similarity between two pairs of words is the Latent Relational Analysis (LRA) [Turney, 2006]. Given a set of word-pairs, LRA first extracts n-grams of words from the contexts (sentences) in which two words co-occur. For example, given the two words *ostrich* and *bird*, lexical patterns such as ***X** is a large **Y*** and ***X** is a **Y*** are extracted. Next, a co-occurrence matrix is built in which the rows correspond to word-pairs and the columns correspond to lexical patterns. An element h_{ij} of $\mathbf{H}$ denotes the number of times the i-th word-pair (a_i, b_i) co-occurs with the j-th lexical pattern p_j in a text corpus. Next, singular value decomposition (SVD) is performed on $\mathbf{H}$ to reduce the data sparseness, and the largest l left singular vectors are selected to build a matrix $\mathbf{H}'$ with l columns. Finally, the relational similarity between word pairs (a_i, b_i) and (a_j, b_j) is measured by the cosine of the angle between the i-th and j-th row vectors in $\mathbf{H}'$. LRA reports an accuracy of 56.1% on a benchmark dataset containing 374 SAT verbal analogy questions.

Veale [Veale, 2004] proposed a relational similarity measure based on the taxonomic similarity in the WordNet [Miller, 1995]. He evaluates the quality of a candidate analogy *a:b::c:d* by comparing the semantic relations that corresponds to the paths in the WordNet that connects a to b and c to d. This approach achieves an accuracy of 43% on SAT verbal analogies.

Bollegala et al. [Bollegala *et al.*, 2009] proposed a supervised approach for learning a Mahalanobis distance metric between semantic relations. Similar to LRA, a word pair (a, b) is represented by a vector of lexical patterns that co-occur with a and b. Next, a sequential pattern clustering algorithm is used to cluster semantically similar lexical patterns. Because a single semantic relation can be expressed using multiple lexical-syntactic patterns, this clustering step groups semantically similar lexical patterns together, and reduces the data sparseness. Finally, an information-theoretic metric learning algorithm [Davis *et al.*, 2007] is used to compute a Mahalanobis distance matrix over the clusters, reporting an accuracy of 51.1% on the SAT verbal analogies.

Latent Relational Mapping Engine (LRME) [Turney, 2008] finds a bijective mapping between two lists of word pairs. To represent the semantic relations between two words, LRME extracts numerous lexical patterns in which those two words co-occur. Then LRA is used to measure the relational similarity between two pairs of words each taken from one of the lists. All possible pairwise combinations of word pairs are considered and the mapping that yields the highest average relational similarity is selected.

All of the above-mentioned prior work in relational similarity consider only the analogies that exist between two *pairs* of words, whereas the proposed method considers tuples that contain three words. Moreover, given an entity-relation graph, we extract all analogous tuple pairs, whereas the prior work on relational similarity assume that the two word pairs between which we must compute relational similarity are given.

Entity-Relation Graph

Much of the real-world knowledge that we have about named entities such as people, products, or organizations can be concisely expressed using the semantic relations that exist among those named entities. Consequently, we consider an *entity-relation* graph as the input for the proposed method for extracting analogous tuples of entities. In an entity-relation graph, each vertex corresponds to a unique entity, and the edge that connects two vertices represents the semantic relation that exists between the corresponding entities. Moreover, an edge can be both directional and/or weighted according to the strength of the semantic relation that is represented by that edge. Numerous entity-relation graphs that contain millions of entities have been created in prior work in information extraction such as YAGO [Suchanek *et al.*, 2007], Freebase [Bollacker *et al.*, 2008], WikiNet [Nastase *et al.*, 2010], WikiTaxonomy [Ponzetto and Strube, 2007], DBpedia [Auer *et al.*, 2007], and PATTY [Nakashole *et al.*, 2012].

Although those work differ in the methods they use for disambiguating entities, extracting and representing relations, their outputs can be recognized as entity-relation graphs. In particular, the representations used to denote a semantic relation differs from one entity-relation graph to another. For example, one can use n-grams of words from the contexts in which two entities co-occur as lexical patterns to represent the semantic relation between those two entities [Hearst, 1992]. Lexical patterns can be generalized using part-of-speech tag sequences or extended using dependency relations [Snow *et al.*, 2005]. Because a single semantic relation can be expressed in a text by multiple patterns, some approaches cluster the extracted patterns to group semantically similar patterns [Nakashole *et al.*, 2012; Lin and Pantel, 2001]. Our method does not assume a particular relation representation method and can potentially extract analogous tuples from any of the entity-relation graphs created using the above-mentioned prior work. Here onwards, we use the term *pattern* to collectively refer to any of the above-mentioned relation representation methods.

We formally define an entity-relation graph $\mathcal{G} = (\mathcal{V}, \mathcal{P}, w)$ as a labeled, weighted, and directional graph where $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices (corresponding to n unique entities), $\mathcal{P} = \{p_1, p_2, \ldots, p_m\}$ is the set of patterns, and the weight function $w(v, u, p)$ indicates the strength of the pattern p that flows from vertex v to u (i.e. direction is $v \rightarrow u$). There have been numerous methods proposed for measuring the strength of a semantic relation represented by a pattern that co-occurs with two entities. For example, Turney [Turney, 2006] uses positive pointwise mutual information for this purpose in LRA. Because semantic relations are directional in general, $w(v, u, p) \neq w(u, v, p)$. Moreover, we assume that the weight function w is non-negative.

Mining for Candidate Tuple Pairs

Given an entity-relation graph $\mathcal{G}$, Algorithm 1 extracts a set of potential candidate pairs of analogous tuples. In the next Section, we propose numerous kernel functions to exactly compute the degree of analogy between two given tuples.

Algorithm 1 Candidate Tuple Pair Extraction.

Input: An entity-relation graph $\mathcal{G} = (\mathcal{V}, \mathcal{P}, w)$.
Output: A set $\mathcal{C}$ of candidate pairs of tuples.

```
1: S ← {}
2: for v_i ∈ V do
3:     for v_j, v_k ∈ N(v_i) do
4:         S ← S + {(v_i, v_j, v_k)}
5:     end for
6: end for
7: C ← {}
8: for (v_i, v_j, v_k) ∈ S do
9:     for p ∈ g(v_i, v_j) ∪ g(v_j, v_k) ∪ g(v_k, v_i) do
10:        for v_i', v_j', v_k' ∈ H(p) do
11:            if (v_i', v_j', v_k') ∈ S then
12:                C ← C + {(v_i', v_j', v_k')}
13:            end if
14:        end for
15:    end for
16: end for
17: return C
```

In Lines 1-6 in Algorithm 1, for each vertext v_i, we create tuples from vertices that are directly connected to v_i. This step can be conducted efficiently by representing the graph as lists of *neighbours* $\mathcal{N}(v)$ for each vertex v as follows:

$$\{u \mid u \in \mathcal{V}, \exists p \in \mathcal{P}, (w(v, u, p) \neq 0) \vee (w(u, v, p) \neq 0)\}.$$

Next, in Lines 7-15, for each tuple (v_i, v_j, v_k) generated in the previous step, we identify other tuples $(v_{i'}, v_{j'}, v_{k'})$ that share at least one pattern in common with (v_i, v_j, v_k). We define $g(v, u)$, the set of patterns that co-occur with either v or u, as follows:

$$g(v, u) = \{p \mid p \in \mathcal{P}, (w(v, u, p) \neq 0) \vee (w(u, v, p) \neq 0)\}.$$

Because patterns represent semantic relations we can safely filter-out tuple pairs that do not share any semantic relations in common, thus unlikely to be analogous. This filtering step reduces the number of tuples that we must compare in the subsequent processing. To speed up the computation, we build an inverted index $\mathcal{H}$ between patterns and entities such that $\mathcal{H}(p)$ returns the set of entities that co-occur with a pattern p and is defined as follows:

$$\{u \mid u \in \mathcal{V}, \exists v \in \mathcal{V}, (w(v, u, p) \neq 0) \vee (w(u, v, p) \neq 0)\}.$$

The time complexity of enumerating triangles is bounded by $\sum_{v \in \mathcal{V}} d(v)^2$, where $d(v)$ is the degree of a vertex v. In power-law graphs, $\sum_{v \in \mathcal{V}} d(v)^2$ is approximately equal to $n d_{\max}^{3-\gamma}$ [Boccaletti *et al.*, 2006], where $d_{\max}$ is the maximum degree and γ is the power coefficient. Moreover, $d_{\max}$ is approximately equal to $n^{1/(\gamma-1)}$ [Newman, 2003], for which the complexity $n^{2/(\gamma-1)}$ follows and is linear in practice.

Tuple-Feature Kernels

To measure the degree of analogy between two tuples (A, B, C) and (X, Y, Z), we define several kernel functions. Let us denote the set of features representing the vertex A by $\mathcal{T}(A)$. Moreover, the weight associated with a feature

$x \in \mathcal{T}(A)$ is denoted by $\theta(x, \mathcal{T}(A))$. For example, the category of an entity (e.g. musician, politician, software company etc.) can be a feature of the vertex represented by that entity. By assigning a weight to a feature representing an entity, we can represent the degree to which a certain entity belongs to a category. Moreover, let us denote the set of features that represent an edge, e_{AB}, connecting the two vertices A and B by $\mathcal{S}(A, B)$. For example, we can use the set of lexical patterns that co-occur with the two entities corresponding to vertices A and B as the features representing the edge e_{AB}. Then, tuple kernel functions, $k((A, B, C), (X, Y, Z))$, between the two tuples (A, B, C) and (X, Y, Z) can be defined in several ways as will be discussed next.

Vertex-Feature Kernels

The simplest form of the tuple kernels use only the features assigned to the individual vertices of each tuple. We define a series of kernel functions based on the set of overlapping features between the disjunctions of the individual vertex feature sets as follows:

$$k_J((A, B, C), (X, Y, Z)) = \text{Jaccard}(\mathcal{T}(\cup_{ABC}), \mathcal{T}(\cup_{XYZ})),$$

$$k_O((A, B, C), (X, Y, Z)) = \text{Overlap}(\mathcal{T}(\cup_{ABC}), \mathcal{T}(\cup_{XYZ})),$$

$$k_D((A, B, C), (X, Y, Z)) = \text{Dice}(\mathcal{T}(\cup_{ABC}), \mathcal{T}(\cup_{XYZ})).$$

Here, to reduce the cluttering of the notation, we define the disjunction of the vertex feature sets corresponding to three vertices A, B, and C as, $\mathcal{T}(\cup_{ABC}) = \mathcal{T}(A) \cup \mathcal{T}(B) \cup \mathcal{T}(C)$.

Given, two fuzzy-sets $\Psi = \{(\psi, \theta(\psi, \Psi))\}$, and $\Lambda = \{(\lambda, \theta(\lambda, \Lambda))\}$, the Jaccard and Overlap coefficients between those fuzzy-sets can be computed as:

$$\text{Jaccard}(\Psi, \Lambda) = \frac{|\Psi \cap \Lambda|}{|\Psi \cup \Lambda|}, \quad (1)$$

$$\text{Overlap}(\Psi, \Lambda) = \frac{|\Psi \cap \Lambda|}{\min(|\Psi|, |\Lambda|)}, \quad (2)$$

Where,

$$|\Psi| = \sum_{(\psi, \theta(\psi, \Psi)) \in \Psi} \theta(\psi, \Psi),$$

$$|\Psi \cap \Lambda| = \sum_{\substack{(\psi, \theta(\psi, \Psi)) \in \Psi \\ (\lambda, \theta(\lambda, \Lambda)) \in \Lambda}} \min(\theta(\psi, \Psi), \theta(\lambda, \Lambda)),$$

$$|\Psi \cup \Lambda| = \sum_{\substack{(\psi, \theta(\psi, \Psi)) \in \Psi \\ (\lambda, \theta(\lambda, \Lambda)) \in \Lambda}} \max(\theta(\psi, \Psi), \theta(\lambda, \Lambda)).$$

Note that it is possible to use other set overlap measures besides the above two to create kernel functions. However, some set overlap measures such as the Dice coefficient has a monotonous relationship with the Jaccard coefficient and are redundant. If there are no weights assigned to the attributes of entities in an entity-relation graph, then we can set all θ values to 1, in which case the above-mentioned kernel functions resort to crisp sets instead of fuzzy-sets.

Edge-Feature Kernels

We define a series of kernel functions based on the features assigned to the edges that exist between pairs of entities in a tuple. First, we define the composition $\mathcal{S}(A, B, C)$, of the three edge feature sets, $\mathcal{S}(A, B)$, $\mathcal{S}(B, C)$, and $\mathcal{S}(C, A)$ as follows:

$$\mathcal{S}(A, B, C) = \mathcal{S}(A, B) \odot \mathcal{S}(B, C) \odot \mathcal{S}(C, A). \quad (3)$$

Here, the operator $\odot$ is defined for two fuzzy-sets Ψ and Λ to give $\Psi \odot \Lambda$ as follows:

$$\{(x, \theta(x, \Psi) \odot \theta(x, \Lambda)) \mid (x, \theta(x, \Psi)) \in \Psi, (x, \theta(x, \Lambda)) \in \Lambda\}$$

We consider addition (+), multiplication (x), minimum (min), and maximum (max) as the candidate operators for $\odot$. We represent each edge-feature set as a fuzzy set in which the membership value of a pattern $p \in \mathcal{P}$ that co-occurs between two entities A and B is given by,

$$\theta(p, \mathcal{S}(A, B)) = w(A, B, p).$$

For example, lexical patterns can be used as edge features to represent the semantic relation that exists between two entities. However, a single semantic relation can be represented using more than one lexical pattern. For example, the ACQUISITION relation that exists between two companies can be represented by ***X is acquired by Y*** as well as ***Y purchased X for NUM dollars***, where *NUM* is some numerical value. On the other hand, not all lexical-syntactic patterns are equally representative of a semantic relation. In our previous example, the verb *acquisition* is more commonly used to describe an ACQUISITION relation between two companies than the verb *purchase*. Therefore, the lexical pattern, ***X is acquired by Y***, is more representative of the ACQUISITION relation between two companies than the lexical pattern, ***Y purchased X for NUM dollars***. Therefore, it is desirable to *softly* assign lexical patterns to edges as features. For example, we can set $\theta(p, \mathcal{S}(A, B))$ to the number of co-occurrences of the two entities A and B with pattern p.

Once we have computed $\mathcal{S}(A, B, C)$ and $\mathcal{S}(X, Y, Z)$ for two tuples (A, B, C) and (X, Y, Z) using Eq. 3, we define two variants $k_J^{\odot}((A, B, C), (X, Y, Z))$ and $k_O^{\odot}((A, B, C), (X, Y, Z))$ of edge-feature kernels with each $\odot$ operator using the Jaccard (Eq. 1) and Overlap (Eq. 2) coefficients given respectively by $\text{Jaccard}(\mathcal{S}(A, B, C), \mathcal{S}(X, Y, Z))$ and $\text{Overlap}(\mathcal{S}(A, B, C), \mathcal{S}(X, Y, Z))$.

Note that we have 8 edge-feature kernels in total with the four operator types $(+, \times, \min, \max)$ for $\odot$ and the two set overlap measures: Jacccard and Overlap coefficients.

Composite Kernels

When measuring the degree of analogy between two tuples, it is desirable to consider both vertex features as well as edge features. However, the above-described vertex-feature kernels and edge-feature kernels individually capture only one type of features. To incorporate both vertex-features and edge-features in the analogy measurement, we combine vertex-feature kernels and edge-feature kernels via kernel composition. Given a vertex-feature kernel k_v and an edge-feature

kernel k_e, we define their composite kernel k_c as the product of the two kernels as follows:

$$k_c((A,B,C),(X,Y,Z)) = k_v((A,B,C),(X,Y,Z)) \times k_e((A,B,C),(X,Y,Z)).$$

Note that the normalization requirement, $k((A,B,C),(A,B,C)) = 1$, is satisfied by all vertex-feature kernels, edge-feature kernels as well as their composite kernels. We combine the two vertex-feature kernels with the eight edge-feature kernels to compute 16 composite kernels. Although there exist numerous other operators besides the product to construct composite kernels from two atomic kernel functions [Shawe-Taylor and Cristianni, 2004], we differ the task of conducting an exhaustive search in the composite kernel space to a future study.

Datasets

We used the PATTY dataset [Nakashole *et al.*, 2012] to create an entity-relation graph. PATTY is a large resource of textual patterns that denote numerous binary relations between entities. In PATTY, the patterns are semantically typed based on categories and organized into a subsumption taxonomy. For example, the pattern **[singer]** *sings* **[song]**, is typed using Wikipedia categories, **singer** and **song**.

We used the PATTY dataset created from the English edition of Wikipedia, which contains about 3.8 million articles (as of June 21, 2011). To create an entity-relation graph from this dataset, first, we create a vertex from each unique entity that appear in the dataset. Next, we create a directional edge between two vertices, if there exists a pattern in PATTY that co-occurs with the two entities that correspond to the two vertices. We use the pattern confidence scores computed in PATTY as the weights for the edges. The entity-relation graph that we create from PATTY contains $749,364$ vertices, and $3,862,331$ connections, corresponding to $350,569$ unique patterns. We use this entity-relation graph for the experiments described in this paper.

Algorithm 1 extracts $171,607$ candidate tuple pairs from this graph. Next, we use numerous kernels described in the previous sections to identify analogous pairs of tuples. On an Intel Xeon 2.9GHz 32 core 256GB RAM machine, it takes less than 10 seconds to process the entire entity-relation graph with a given kernel, which demonstrates the efficiency of our method.

Unlike for word-pair analogies, for which there exists the SAT word-analogy questions benchmark dataset, there is no gold standard benchmark for evaluating analogous tuples. Turney [Turney, 2008] proposed a dataset consisting 20 pairs of word lists, in which there exists a bijective relational mapping between each pair of lists. However, this dataset is not suitable for evaluating our task because we must evaluate whether two tuples are analogous instead of finding a bijective mapping between the tuples. Conversely, if two tuples are not analogous, then there might not exist a bijective relational mapping between those tuples.

Consequently, to evaluate the proposed method, we create a gold standard dataset of analogous tuples. First, we randomly select a source tuples from the set of extracted tuples by Algorithm 1. Next, for each source tuple, we list the candidate tuples that share at least two patterns with it. Three human annotators compared each seed tuple and its candidate list and marked the tuples that they considered to be analogous to the source tuple. In the case where the annotators were not familiar with the entities in a tuple, they are instructed to gather additional information from the Web to determine the semantic relations that exist, if any, between two entities. Finally, for each source tuple we randomly select a tuple that is marked as being analogous to the source tuple by at least two of the three annotators. The final dataset contains 30 pairs of analogous tuples.

Experiments and Results

We consider that it is appropriate to evaluate the proposed method in a retrieval task for two reasons. First, in our task, we must correctly identify the analogous tuples to a given tuple among a large number of non-analogous candidate tuples. This scenario resembles Web search, where we must identify a small number of relevant documents to a given user query among a vast number of irrelevant documents. Kernel functions that assign high similarity scores to tuples that are analogous are desirable. We enforce this condition by limiting the evaluation to the top ranked tuples by a kernel function. Second, precise measurement of recall is difficult in our task because it is labour intensive and costly to go through the entire set of tuples extracted from a large entity-relation graph to identify all pairs of analogous tuples.

For each source tuple in our gold standard, we measure the degree of analogy between it and the remainder of the tuples extracted by Algorithm 1 using a particular kernel function, and rank in the descending order of their similarity scores. We evaluate the top 10 ranked candidates using two popular evaluation measures used in the information retrieval community: Precision@10, and Mean Reciprocal Rank (MRR) [Manning *et al.*, 2008]. Precision@10 is the ratio between the number of analogous tuples found among the top 10 ranked tuples retrieved for a source tuple. MRR is the reciprocal of the rank assigned to the most analogous tuple to a source tuple. Both of those measures are computed for each source tuple in the gold standard and then averaged. Unlike Precision@10, MRR is sensitive to the rank assigned to analogous tuples. Because in the gold standard we assign only a single analogous tuple for each source tuple, Precision@10 can be seen as a measure of accuracy indicating the percentage of source tuples for which we can find their analogous counterparts among the top 10 similar tuples.

In Table 1, we use symbols V, E, and C respectively to denote vertex-feature kernels, edge-feature kernels, and composite kernels. For edge-feature kernels, we indicate the operator $\odot$ and the set overlap measure within brackets. For composite kernels, we show the two constituent kernels used for the composition. We extend LRA, a state-of-the-art method word-pair relational similarity measure, to measure the degree of analogy between two tuples as follows. Given two tuples (A,B,C) and (X,Y,Z), using LRA we measure the relational similarity between all possible combinations of entity pairs between the two tu-

Table 1: Performance for retrieving analogous tuples.

Method	Precision@10	MRR
LRA [Turney, 2006]	0.4666	0.1879
V: Jaccard	0	0
V: Overlap	0	0
E: (+, Jaccard)	0.4	0.1474
E: (+, Overlap)	0.2	0.04841
E: (max, Jaccard)	0.4666	0.2127
E: (max, Overlap)	0.3333	0.1261
E: ($\times$, Jaccard)	0	0
E: ($\times$, Overlap)	0	0
E: (min, Jaccard)	0	0
E: (min, Overlap)	0	0
C: E(+, Jaccard), V:Jaccard	0.6	0.3817
C: E(+, Jaccard) V:Overlap	0.5333	0.2411
C: E(+, Overlap) V:Jaccard	0.4666	0.2295
C: E(+, Overlap) V:Overlap	0.4	0.2266
C: E(max, Jaccard), V:Jaccard	**0.6**	**0.4688**
C: E(max, Jaccard), V:Overlap	0.6	0.3077
C: E(max, Overlap), V:Jaccard	0.4	0.2577
C: E(max, Overlap), V:Overlap	0.5333	0.3088
C: E($\times$, Jaccard), V:Jaccard	0	0
C: E($\times$, Jaccard), V:Overlap	0	0
C: E($\times$, Overlap), V:Jaccard	0	0
C: E($\times$, Overlap), V:Overlap	0	0
C: E(min, Jaccard), V:Jaccard	0	0
C: E(min, Jaccard), V:Overlap	0	0
C: E(min, Overlap), V:Jaccard	0	0
C: E(min, Overlap), V:Overlap	0	0

ples: $((A,B),(X,Y))$, $((B,A),(Y,X))$, $((B,C),(Y,Z))$, $((C,B),(Z,Y))$, $((C,A),(Z,X))$, and $((A,C),(X,Z))$. We consider the average of those relational similarity scores as the degree of analogy between the two tuples. This baseline demonstrates the level of performance we would obtain if we use pairwise relational similarity measure for detecting analogical tuples.

From Table 1, we see that vertex-feature kernels when used alone, does not retrieve any analogous tuples. On the other hand, edge-feature kernels outperform vertex-feature kernels, except when used with $\times$ and min operators. Recall that vertex-feature kernels use *attributes* of entities, whereas edge-feature kernels use patterns that represent the *relations* that exist between entities. According to Structure Mapping Theory [Gentner, 1983], relations, not attributes, are important when finding a mapping between analogous knowledge bases. We believe our empirical results, the superior performance of edge-feature kernels over vertex-feature kernels, support the claim of Structure Mapping Theory. Among the four operators used in edge-feature kernels, max is the best, followed by $+$. Because $\times$ and min operators prefer the weaker features (i.e. patterns with smaller weights) over the stronger features inside a tuple, those operators fail to identify analogous tuples.

Interestingly, when we combine both types of kernels, the performance increases. The best performance is obtained by **C: E(max, Jaccard), V:Jaccard**, which combines the vertex-feature kernel with Jaccard coefficient as the set overlap measure, and the edge-feature kernel that uses the max operator with Jaccard coefficient as the set overlap measure. Moreover, this composite kernel significantly outperforms the LRA baseline (Binomial exact test with significance level of 0.05).

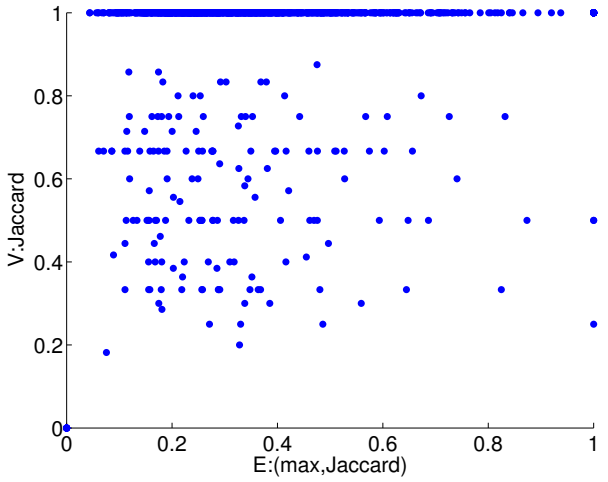

Figure 1: Correlation plot between the best individual vertex-feature and edge-feature kernels.

To further study the improved performance by composite kernels, we randomly select 1000 tuples from our entity-relation graph and compute vertex and edge feature kernels between those tuples. Figure 1 shows those scores for the constituents of the best performing composite kernel (i.e. C: E(max, Jaccard), V:Jaccard). From Figure 1, we see that vertex-feature and edge-feature kernels measure very different aspects of tuples. For example, for numerous pairs of tuples for which vertex-feature kernel assigns perfect scores, we must resort to edge-feature kernels to discriminate the correct analogous tuple pairs. Moreover, the Pearson correlation coefficient between the two kernels is low at 0.1356. Similar trends were observed between other combinations of vertex-feature and edge-feature kernels as well. Therefore, the combination of vertex-feature and edge-feature kernels can be considered as providing mutually exclusive information, which justifies our kernel composition approach.

Conclusion

Given an entity-relation graph as the input, we proposed a method to extract analogous pairs of tuples that contain three entities. First, we proposed an algorithm to efficiently generate candidate pairs of tuples. Second, we proposed two types of kernel functions based on vertex-features and edge-features to measure the degree of analogy between two tuples. Moreover, we derived composite kernels from vertex and edge feature kernels. Our experiments using a real-world entity-relation graph showed that the proposed kernel composition approach significantly outperforms the current state-of-the-art pairwise relational similarity measure, extended to cover tuples with three entities. Moreover, we show that edge-features play an important role when detecting analogous tuples. In future, we plan to extend the proposed method to handle tuples with $n(>3)$ entities, thereby identifying analogous subgraphs within an entity-relation graph.

References

[Auer *et al.*, 2007] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: a nucleus for a web of open data. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, ISWC'07/ASWC'07, pages 722–735, Berlin, Heidelberg, 2007. Springer-Verlag.

[Boccaletti *et al.*, 2006] S. Boccaletti, V. Latora, Y. Monreno, M. Chavez, and D.U. Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424:175 – 308, 2006.

[Bollacker *et al.*, 2008] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD'08*, pages 1247 – 1250, 2008.

[Bollegala *et al.*, 2009] Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. Measuring the similarity between implicit semantic relations from the web. In *WWW 2009*, pages 651 – 660, 2009.

[Davis *et al.*, 2007] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *ICML'07*, pages 209–216, 2007.

[Duc *et al.*, 2011] Nguyen Tuan Duc, Danushka Bollegala, and Mitsuru Ishizuka. Cross-language latent relational search: Mapping knowledge across languages. In *AAAI'11*, pages 1237 – 1242, 2011.

[Falkenhainer *et al.*, 1989] B. Falkenhainer, K.D. Forbus, and D. Gentner. Structure mapping engine: Algorithm and examples. *Artificial Intelligence*, 41:1–63, 1989.

[Gentner and Gentner, 1983] Dedre Gentner and Donald R. Gentner. *Flowing waters or teeming crowds: Mental models of electricity*, chapter 6, pages 99 – 129. Lawrence Erlbaum Associates, Hillsdale, NJ, 1983.

[Gentner, 1983] Dedre Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7:155 – 170, 1983.

[Hearst, 1992] M.A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proc. of 14th COLING*, volume 2, pages 539–545, 1992.

[Kotovsky and Gentner, 1996] Laura Kotovsky and Dedre Gentner. Comparison and categorization in the development of relational similarity. *Child Development*, 67:2797–2822, 1996.

[Lin and Pantel, 2001] Dekang Lin and Patrick Pantel. Dirt - discovery of inference rules from text. In *KDD 2001*, pages 323–328, 2001.

[Manning *et al.*, 2008] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[Miller, 1995] George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39 – 41, November 1995.

[Nakashole *et al.*, 2012] Ndapandula Nakashole, Gerhard Weikum, and Fabian M. Suchanek. Patty: A taxonomy of relational patterns with semantic types. In *EMNLP'12*, pages 1135 – 1145, 2012.

[Nastase *et al.*, 2010] Vivi Nastase, Michael Strube, Benjamin Boerschinger, Caecilia Zirn, and Anas Elghafari. Wikinet: A very large scale multi-lingual concept network. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010. European Language Resources Association (ELRA).

[Newman, 2003] M. E. J. Newman. The structure and function of complex networks. *SIAM REVIEW*, 45:167 – 256, 2003.

[Ponzetto and Strube, 2007] Simone Paolo Ponzetto and Michael Strube. Deriving a large scale taxonomy from wikipedia. In *AAAI'07*, pages 1440 – 1445, 2007.

[Shawe-Taylor and Cristianni, 2004] John Shawe-Taylor and Nello Cristianni. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[Snow *et al.*, 2005] R. Snow, D. Jurafsky, and A.Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *Proc. of Advances in Neural Information Processing Systems (NIPS) 17*, pages 1297–1304, 2005.

[Suchanek *et al.*, 2007] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge unifying wordnet and wikipedia. In *WWW'07*, pages 697 – 706, 2007.

[Turney *et al.*, 2003] P.D. Turney, M.L. Littman, J. Bigham, and V. Shnayder. Combining independent modules to solve multiple-choice synonym and analogy problems. In *Proc. of RANLP'03*, pages 482–486, 2003.

[Turney, 2005] P.D. Turney. Measuring semantic similarity by latent relational analysis. In *Proc. of IJCAI'05*, pages 1136–1141, 2005.

[Turney, 2006] P.D. Turney. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416, 2006.

[Turney, 2008] P. D. Turney. The latent relational mapping engine: Algorithm and experiments. *Journal of Artificial Intelligence Research*, 33:615–655, 2008.

[Veale and Keane, 2003] T. Veale and M. T. Keane. The competence of structure mapping on hard analogies. In *Proc. of IJCAI'03*, 2003.

[Veale, 2004] T. Veale. Wordnet sits the sat: A knowledge-based approach to lexical analogy. In *Proc. of ECAI'04*, pages 606–612, 2004.