Incorporating Vector Space Similarity in Random Walk Inference over Knowledge Bases

Matt Gardner
Carnegie Mellon University
mg1@cs.cmu.edu

Partha Talukdar[*]
Indian Institute of Science
ppt@serc.iisc.in

Jayant Krishnamurthy
Carnegie Mellon University
jayantk@cs.cmu.edu

Tom Mitchell
Carnegie Mellon University
tom@cs.cmu.edu

Abstract

Much work in recent years has gone into the construction of large knowledge bases (KBs), such as Freebase, DBPedia, NELL, and YAGO. While these KBs are very large, they are still very incomplete, necessitating the use of inference to fill in gaps. Prior work has shown how to make use of a large text corpus to augment random walk inference over KBs. We present two improvements to the use of such large corpora to augment KB inference. First, we present a new technique for combining KB relations and surface text into a single graph representation that is much more compact than graphs used in prior work. Second, we describe how to incorporate vector space similarity into random walk inference over KBs, reducing the feature sparsity inherent in using surface text. This allows us to combine distributional similarity with symbolic logical inference in novel and effective ways. With experiments on many relations from two separate KBs, we show that our methods significantly outperform prior work on KB inference, both in the size of problem our methods can handle and in the quality of predictions made.

1 Introduction

Much work in recent years has gone into the construction of large knowledge bases, either by collecting contributions from many users, as with Freebase (Bollacker et al., 2008) and DBPedia (Mendes et al., 2012), or automatically from web text or other resources, as done by NELL (Carlson et al., 2010) and YAGO (Suchanek et al., 2007). These knowledge bases contain millions of real-world entities and relationships between them. However, even though they are very large, they are still very incomplete, missing large fractions of possible relationships between common entities (West et al., 2014). Thus the task of *inference* over these knowledge bases, predicting new relationships simply by examining the knowledge base itself, has become increasingly important.

A promising technique for inferring new relation instances in a knowledge base is random walk inference, first proposed by Lao and Cohen (2010). In this method, called the Path Ranking Algorithm (PRA), the knowledge base is encoded as a graph, and random walks are used to find paths that connect the source and target nodes of relation instances. These paths are used as features in a logistic regression classifier that predicts new instances of the given relation. Each path can be viewed as a horn clause using knowledge base relations as predicates, and so PRA can be thought of as a kind of discriminatively trained logical inference.

One major deficiency of random walk inference is the connectivity of the knowledge base graph—if there is no path connecting two nodes in the graph, PRA cannot predict any relation instance between them. Thus prior work has introduced the use of a text corpus to increase the connectivity of the graph used as input to PRA (Lao et al., 2012; Gardner et al., 2013). This approach is not without its own problems, however. Whereas knowledge base relations are semantically coherent and different relations have distinct meanings, this is not

[*] Research carried out while at the Machine Learning Department, Carnegie Mellon University.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 397–406,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

true of surface text. For example, “The Nile *flows through* Cairo” and “The Nile *runs through* Cairo” have very similar if not identical meaning. Adding a text corpus to the inference graph increases connectivity, but it also dramatically increases feature sparsity.

We introduce two new techniques for making better use of a text corpus for knowledge base inference. First, we describe a new way of incorporating the text corpus into the knowledge base graph that enables much more efficient processing than prior techniques, allowing us to approach problems that prior work could not feasibly solve. Second, we introduce the use of vector space similarity in random walk inference in order to reduce the sparsity of surface forms. That is, when following a sequence of edge types in a random walk on a graph, we allow the walk to follow edges that are semantically similar to the given edge types, as defined by some vector space embedding of the edge types. If a path calls for an edge of type “flows through”, for example, we accept other edge types (such as “runs through”) with probability proportional to the vector space similarity between the two edge types. This lets us combine notions of distributional similarity with symbolic logical inference, with the result of decreasing the sparsity of the feature space considered by PRA. We show with experiments using both the NELL and Freebase knowledge bases that this method gives significantly better performance than prior approaches to incorporating text data into random walk inference.

2 Graph Construction

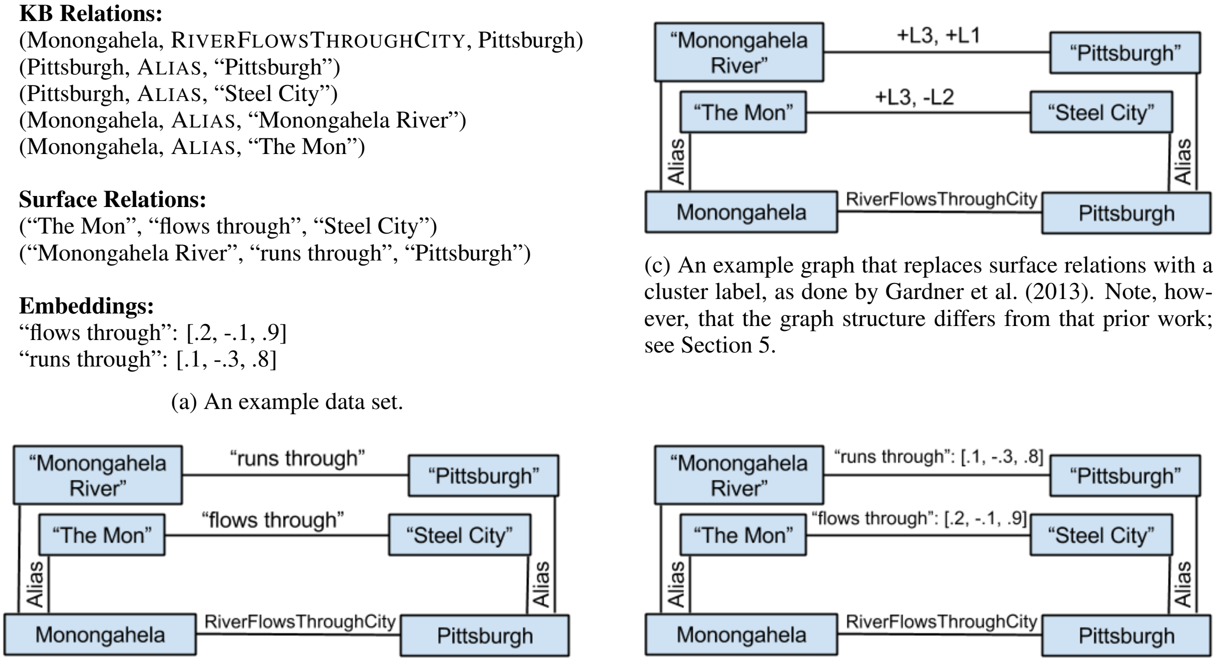
Our method for knowledge base inference, described in Section 3, performs random walks over a graph to obtain features for a logistic regression classifier. Prior to detailing that technique, we first describe how we produce a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E}, \mathcal{R})$ from a set of knowledge base (KB) relation instances and a set of surface relation instances extracted from a corpus. Producing a graph from a knowledge base is straightforward: the set of nodes $\mathcal{N}$ is made up of the entities in the KB; the set of edge types $\mathcal{R}$ is the set of relation types in the KB, and the typed edges $\mathcal{E}$ correspond to relation instances from the KB, with one edge of type r connecting entity nodes for each (n_1, r, n_2) triple in the KB. Less straightforward is how to construct a graph from a corpus, and how to connect that graph to the KB graph. We describe our methods for each of those below.

To create a graph from a corpus, we first preprocess the corpus to obtain a collection of *surface relations*, such as those extracted by open information extraction systems like OLLIE (Mausam et al., 2012). These surface relations consist of a pair of noun phrases in the corpus, and the verb-like connection between them (either an actual verb, as done by Talukdar et al. (2012), a dependency path, as done by Riedel et al. (2013), or OpenIE relations (Mausam et al., 2012)). The verb-like connections are naturally represented as edges in the graph, as they have a similar semantics to the knowledge base relations that are already represented as edges. We thus create a graph from these triples exactly as we do from a KB, with nodes corresponding to noun phrase types and edges corresponding to surface relation triples.

So far these two subgraphs we have created are entirely disconnected, with the KB graph containing nodes representing entities, and the surface relation graph containing nodes representing noun phrases, with no edges between these noun phrases and entities. We connect these two graphs by making use of the ALIAS relation in the KB, which links entities to potential noun phrase referents. Each noun phrase in the surface relation graph is connected to those entity nodes which the noun phrase can possibly refer to according to the KB. These edges are not the output of an entity linking system, as done by Lao et al. (2012), but express instead the notion that the noun phrase *can* refer to the KB entity. The use of an entity linking system would certainly allow a stronger connection between noun phrase nodes and entity nodes, but it would require much more preprocessing and a much larger graph representation, as each mention of each noun phrase would need its own node, as opposed to letting every mention of the same noun phrase share the same node. This graph representation allows us to add tens of millions of surface relations to a graph of tens of millions of KB relations, and perform all of the processing on a single machine.

As will be discussed in more detail in Section 4, we also allow edge types to optionally have an associated vector that ideally captures something of the semantics of the edge type.

Figure 1 shows the graph constructions used in our experiments on a subset of KB and surface re-



KB Relations:
(Monongahela, RIVERFLOWSTHROUGHCITY, Pittsburgh)
(Pittsburgh, ALIAS, “Pittsburgh”)
(Pittsburgh, ALIAS, “Steel City”)
(Monongahela, ALIAS, “Monongahela River”)
(Monongahela, ALIAS, “The Mon”)

Surface Relations:
(“The Mon”, “flows through”, “Steel City”)
(“Monongahela River”, “runs through”, “Pittsburgh”)

Embeddings:
“flows through”: [.2, -.1, .9]
“runs through”: [.1, -.3, .8]

(a) An example data set.

(c) An example graph that replaces surface relations with a cluster label, as done by Gardner et al. (2013). Note, however, that the graph structure differs from that prior work; see Section 5.

(b) An example graph that combines a KB and surface relations.

(d) An example graph that uses vector space representations of surface edges, as introduced in this paper.

Figure 1: Example graph construction as used in the experiments in this paper. A graph using only KB edges is simply a subset of these graphs containing only the RIVERFLOWSTHROUGHCITY edge, and is not shown.

lations. Note that Figures 1b and 1c are shown as rough analogues of graphs used in prior work (described in more detail in Section 5), and we use them for comparison in our experiments.

3 The Path Ranking Algorithm

We perform knowledge base inference using the Path Ranking Algorithm (PRA) (Lao and Cohen, 2010). We begin this section with a brief overview of PRA, then we present our modification to the PRA algorithm that allows us to incorporate vector space similarity into random walk inference.

PRA can be thought of as a method for exploiting local graph structure to generate non-linear feature combinations for a prediction model. PRA generates a feature matrix over pairs of nodes in a graph, then uses logistic regression to classify those node pairs as belonging to a particular relation.

More formally, given a graph $\mathcal{G}$ with nodes $\mathcal{N}$, edges $\mathcal{E}$, and edge labels $\mathcal{R}$, and a set of node pairs $(s_i, t_i) \in \mathcal{D}$, one can create a connectivity matrix where rows correspond to node pairs and columns correspond to edge lables. PRA *augments* this matrix with additional columns corresponding to *sequences* of edge labels, called *path types*, and changes the cell values from representing the presence of an edge to representing the specificity of the connection that the path type makes between the node pair.

Because the feature space considered by PRA is so large (the set of all possible edge label sequences, with cardinality $\sum_{i=1}^{l} |\mathcal{R}|^i$, assuming a bound l on the maximum path length), the first step PRA must perform is *feature selection*, which is done using random walks over the graph. The second step of PRA is *feature computation*, where each cell in the feature matrix is computed using a constrained random walk that follows the path type corresponding to the feature. We now explain each of these steps in more detail.

Feature selection finds path types π that are likely to be useful in predicting new instances of the relation represented by the input node pairs . These path types are found by performing random walks on the graph $\mathcal{G}$ starting at the source and target nodes in $\mathcal{D}$, recording which paths connect some source node with its target. The edge sequences are ranked by frequency of connecting a source node to a corresponding target node, and the top k are kept.

Feature computation. Once a set of path types is selected as features, the next step of the PRA algorithm is to compute a value for each cell in the feature matrix, corresponding to a node pair and a path type. The value computed is the probability of arriving at the target node of a node pair, given that a random walk began at the source node and was constrained to follow the path type: $\mathrm{p}(t|s, \pi)$.

Once these steps have been completed, the resulting feature matrix can be used with whatever model or learning algorithm is desired; in this and prior work, simple logistic regression has been used as the prediction algorithm.

4 Vector space random walks

Our modifications to PRA are confined entirely to the feature computation step described above; feature selection (finding potentially useful sequences of edge types) proceeds as normal, using the symbolic edge types. When computing feature values, however, we allow a walk to follow an edge that is *semantically similar* to the edge type in the path, as defined by Euclidean distance in the vector space.

More formally, consider a path type π. Recall that π is a sequence of edge types $< e_1, e_2, \ldots, e_l >$, where l is the length of the path; we will use π_i to denote the i^{th} edge type in the sequence. To compute feature values, PRA begins at some node and follows edges of type π_i until the sequence is finished and a target node has been reached. Specifically, if a random walk is at a node n with m outgoing edge types $\{e_1, e_2, \ldots, e_m\}$, PRA selects the edge type from that set which matches π_i, then selects uniformally at random from all outgoing edges of that type. If there is no match in the set, the random walk restarts from the original start node.

We modify the selection of which edge type to follow. When a random walk is at a node n with m outgoing edge types $\{e_1, e_2, \ldots, e_m\}$, instead of selecting only the edge type that matches π_i, we allow the walk to select instead an edge that is close to π_i in vector space. For each edge type at node n, we select the edge with the following probability:

$$p(e_j|\pi_i) \propto \exp(\beta \times v(e_j) \cdot v(\pi_i)),\ \forall j, 1 \leq j \leq m$$

where $v(\cdot)$ is a function that returns the vector representation of an edge type, and β is a spikiness parameter that determines how much weight to give to the vector space similarity. As β approaches infinity, the normalized exponential approximates a delta function on the closest edge type to π_i, in $\{e_1, e_2, \ldots, e_m\}$. If π_i is in the set of outgoing edges, this algorithm converges to the original PRA.

However, if π_i is not in the set of outgoing edge types at a node and all of the edge types are very dissimilar to π_i, this algorithm (with β not close to infinity) will lead to a largely uniform distribution over edge types at that node, and no way for the random walk to restart. To recover the restart behavior of the original PRA, we introduce an additional restart parameter α, and add another value to the categorical distribution before normalization:

$$p(\text{restart}|\pi_i) \propto \exp(\beta * \alpha)$$

When this *restart* type is selected, the random walk begins again, following π_1 starting at the source node. With α set to a value greater than the maximum similarity between (non-identical) edge type vectors, and β set to infinity, this algorithm exactly replicates the original PRA.

Not all edge types have vector space representations: the ALIAS relation cannot have a meaningful vector representation, and we do not use vectors to represent KB relations, finding that doing so was not useful in practice (which makes intuitive sense: KB relations are already latent representations themselves). While performing random walks, if π_i has no vector representation, we fall back to the original PRA algorithm for selecting the next edge.

We note here that when working with vector spaces it is natural to try clustering the vectors to reduce the parameter space. Each path type π is a feature in our model, and if two path types differ only in one edge type, and the differing edge types have very similar vectors, the resultant feature values will be essentially identical for both path types. It seems reasonable that running a simple clustering algorithm over these path types, to reduce the number of near-duplicate features, would improve performance. We did not find this to be the case, however; all attempts we made to use clustering over these vectors gave performance indistinguishable from not using clustering. From this we conclude that the main issue hindering performance when using PRA over these kinds of graphs is one of limited connectivity, not one of too many parameters in the model. Though the feature space considered by PRA is very large, the number of attested features in a real graph is much smaller, and it is this sparsity which our vector space methods address.

5 Related Work

Knowledge base inference. Random walk inference over knowledge bases was first introduced by Lao and Cohen (2010). This work was improved upon shortly afterward to also make use of a large corpus, by representing the corpus as a graph and connecting it to the knowledge base (Lao et al., 2012). Gardner et al. (2013) further showed that replacing surface relation labels with a representation of a latent embedding of the relation led to improved prediction performance. This result is intuitive: the feature space considered by PRA is exponentially large, and surface relations are sparse. The relations "[river] flows through [city]" and "[river] runs through [city]" have near identical meaning, and both should be very predictive for the knowledge base relation RIVERFLOWSTHROUGHCITY. However, if one of these relations only appears in the training data and the other only appears in the test data, neither will be useful for prediction. Gardner et al. (2013) attempted to solve this issue by finding a latent symbolic representation of the surface relations (such as a clustering) and replacing the edge labels in the graph with these latent representations. This makes it more likely for surface relations seen in training data to also be seen at test time, and naturally improved performance.

This representation, however, is still brittle, as it is still a symbolic representation that is prone to mismatches between training and test data. If the clustering algorithm used is too coarse, the features will not be useful, and if it is too fine, there will be more mismatches. Also, verbs that are on the boundaries of several clusters are problematic to represent in this manner. We solve these problems by modifying the PRA algorithm to directly use vector representations of edge types during the random walk inference.

These two prior techniques are the most directly related work to what we present in this paper, and we compare our work to theirs.

Graph construction. In addition to the incorporation of vector space similarity into the PRA algorithm, the major difference between our work and the prior approaches mentioned above is in the construction of the graph used by PRA. We contrast our method of graph construction with these prior approaches in more detail below.

Lao et al. (2012) represent every word of every sentence in the corpus as a node in the graph, with edges between the nodes representing dependency relationships between the words. They then connect this graph to the KB graph using a simple entity linking system (combined with coreference resolution). The resultant graph is enormous, such that they needed to do complex indexing on the graph and use a cluster of 500 machines to perform the PRA computations. Also, as the edges represent dependency labels, not words, with this graph representation the PRA algorithm does not have access to the verbs or other predicative words that appear in the corpus, which frequently express relations. PRA only uses *edge* types as feature components, not *node* types, and so the rich information contained in the words is lost. This graph construction also would not allow the incorporation of vector space similarity that we introduced, as dependency labels do not lend themselves well to vector space representations.

Gardner et al. (2013) take an approach very similar to the one presented in Section 2, preprocessing the corpus to obtain surface relations. However, instead of creating a graph with nodes representing noun phrases, they added edges from the surface relations directly to the entity nodes in the graph. Using the ALIAS relation, as we do, they added an edge between *every possible* concept pair that could be represented by the noun phrases in a surface relation instance. This leads to some nonsensical edges added to the graph, and if the ALIAS relation has high degree (as it does for many common noun phrases in Freebase), it quickly becomes unscalable—this method of graph construction runs out of disk space when attempting to run on the Freebase experiments in Section 6. Also, in conflating entity nodes in the graph with noun phrases, they lose an important distinction that turns out to be useful for prediction, as we discuss in Section 6.4.[1]

[1] Recent notions of "universal schema" (Riedel et al., 2013) also put KB entities and noun phrases into the same conceptual space, though they opt for using noun phrases instead of the KB entities used by Gardner et al. In general this is problematic, as it relies on some kind of entity linking system as preprocessing, and cannot handle common noun references of proper entities without losing information. Our method, and that of Lao et al., skirts this issue entirely by not trying to merge KB entities with noun phrases.

Other related work. Also related to the present work is recent research on programming languages for probabilistic logic (Wang et al., 2013). This work, called ProPPR, uses random walks to locally ground a query in a small graph before performing propositional inference over the grounded representation. In some sense this technique is like a recursive version of PRA, allowing for more complex inferences than a single iteration of PRA can make. However, this technique has not yet been extended to work with large text corpora, and it does not yet appear to be scalable enough to handle the large graphs that we use in this work. How best to incorporate the work presented in this paper with ProPPR is an open, and very interesting, question.

Examples of other systems aimed at reasoning over common-sense knowledge are the CYC project (Lenat, 1995) and ConceptNet (Liu and Singh, 2004). These common-sense resources could easily be incorporated into the graphs we use for performing random walk inference.

Lines of research that seek to incorporate distributional semantics into traditional natural language processing tasks, such as parsing (Socher et al., 2013a), named entity recognition (Passos et al., 2014), and sentiment analysis (Socher et al., 2013b), are also related to what we present in this paper. While our task is quite different from these prior works, we also aim to combine distributional semantics with more traditional methods (in our case, symbolic logical inference), and we take inspiration from these methods.

6 Experiments

We perform both the feature selection step and the feature computation step of PRA using GraphChi, an efficient single-machine graph processing library (Kyrola et al., 2012). We use MALLET's implementation of logistic regression, with both L1 and L2 regularization (McCallum, 2002). To obtain negative evidence, we used a closed world assumption, treating any (source, target) pair found during the feature computation step as a negative example if it was not given as a positive example. We tuned the parameters to our methods using a coarse, manual grid search with cross validation on the training data described below. The parameters we tuned were the L1 and L2 regularization parameters, how many random walks to perform in the feature selection and computation steps of PRA, and spikiness and restart parameters for vector space walks. The results presented were not very sensitive to changes in these parameters.

	NELL	Freebase
Entities	1.2M	20M
Relation instances	3.4M	67M
Total relation types	520	4215
Relation types tested	10	24
Avg. instances/relation	810	200
SVO triples used	404k	28M

Table 1: Statistics of the data used in our experiments.

6.1 Data

We ran experiments on both the NELL and Freebase knowledge bases. The characteristics of these knowledge bases are shown in Table 1. The Freebase KB is very large; to make it slightly more manageable we filtered out relations that did not seem applicable to relation extraction, as well as a few of the largest relations.[2] This still left a very large, mostly intact KB, as can be seen in the table. For our text corpus, we make use of a set of subject-verb-object triples extracted from dependency parses of ClueWeb documents (Talukdar et al., 2012). There are 670M such triples in the data set, most of which are completely irrelevant to the knowledge base relations we are trying to predict. For each KB, we filter the SVO triples, keeping only those which can possibly connect training and test instances of the relations we used in our experiments. The number of SVO triples kept for each KB is also shown in Table 1. We obtained vector space representations of these surface relations by running PCA on the SVO matrix.

We selected 10 NELL relations and 24 Freebase relations for testing our methods. The NELL relations were hand-selected as the relations with the largest number of known instances that had a reasonable precision (the NELL KB is automatically created, and some relations have low precision). We split the known instances of these relations into 75% training and 25% testing, giving on average about 650 training instances and 160 test

[2] We removed anything under /user, /common, /type (except for the relation /type/object/type), /base, and /freebase, as not applicable to our task. We also removed relations dealing with individual music tracks, book editions, and TV epidsodes, as they are very large, very specific, and unlikely to be useful for predicting the relations in our test set.

instances for each relation.

The 24 Freebase relations were semi-randomly selected. We first filtered the 4215 relations based on two criteria: the number of relation instances must be between 1000 and 10000, and there must be no mediator in the relation.[3] Once we selected the relations, we kept all instances of each relation that had some possible connection in the SVO data.[4] This left on average 200 instances per relation, which we again split 75%-25% into training and test sets.

6.2 Methods

The methods we compare correspond to the graphs shown in Figure 1. The KB method uses the original PRA algorithm on just the KB relations, as presented by Lao and Cohen (2010). KB + SVO adds surface relations to the graph (Figure 1b). We present this as roughly analogous to the methods introduced by Lao et al. (2012), though with some significant differences in graph representation, as described in Section 5. KB + Clustered SVO follows the methods of Gardner et al. (2013), but using the graph construction introduced in this paper (Figure 1c; their graph construction techniques would have made graphs too large to be feasible for the Freebase experiments). KB + Vector SVO is our method (Figure 1d).

6.3 Evaluation

As evaluation metrics, we use *mean average precision* (MAP) and *mean reciprocal rank* (MRR), following recent work evaluating relation extraction performance (West et al., 2014). We test significance using a paired permutation test.

The results of these experiments are shown in Table 2 and Table 3. In Table 4 we show average precision for every relation tested on the NELL KB, and we show the same for Freebase in Table 5.

6.4 Discussion

We can see from the tables that KB + Vector SVO (the method presented in this paper) significantly outperforms prior approaches in both MAP and MRR. We believe that this is due to the reduction in feature sparsity enabled by using vector space instead of symbolic representations (as that is the only real difference between KB + Clustered SVO and KB + Vector SVO), allowing PRA to make better use of path types found in the training data. When looking at the results for individual relations in Table 4 and Table 5, we see that KB + Vector SVO outperforms other methods on the majority of relations, and it is a close second when it does not.

Method	MAP	MRR
KB	0.193	0.635
KB + SVO	0.218	0.763
KB + Clustered SVO	0.276	0.900
KB + Vector SVO	**0.301**	**0.900**

Table 2: Results on the NELL knowledge base. The bolded line is significantly better than all other results with $p < 0.025$.

Method	MAP	MRR
KB	0.278	0.614
KB + SVO	0.294	0.639
KB + Clustered SVO	0.326	0.651
KB + Vector SVO	**0.350**	**0.670**

Table 3: Results on the Freebase knowledge base. The bolded line is significantly better than all other results with $p < 0.0002$.

We can also see from the results that mean average precision seems a little low for all methods tested. This is because MAP is computed as the precision of *all* possible correct predictions in a ranked list, where precision is counted as 0 if the correct prediction is not included in the list. In other words, there are many relation instances in our randomly selected test set that are not inferrable from the knowledge base, and the low recall hurts the MAP metric. MRR, which judges the precision of the top prediction for each relation, gives us some confidence that the main issue here is one of recall, as MRR is reasonably high, especially on the NELL KB. As further evidence, if we compute average precision for each query node (instead of for each relation), excluding queries for which the system did not return any predictions, MAP ranges from .29 (KB) to .45 (KB + Vector SVO) on NELL (with around 30% of queries having no prediction), and from .40 (KB) to .49 (KB +

[3] A mediator in Freebase is a reified relation instance meant to handle n-ary relations, for instance /film/performance. PRA in general, and our implementation of it in particular, needs some modification to be well-suited to predicting relations with mediators.

[4] We first tried randomly selecting instances from these relations, but found that the probability of selecting an instance that benefited from an SVO connection was negligible. In order to make use of the methods we present, we thus restricted ourselves to only those that had a possible SVO connection.

Relation	KB	KB + SVO	KB + Clustered SVO	KB + Vector SVO
ActorStarredInMovie	0.000	0.032	0.032	**0.037**
AthletePlaysForTeam	0.200	0.239	0.531	**0.589**
CityLocatedInCountry	0.126	0.169	0.255	**0.347**
JournalistWritesForPublication	0.218	0.254	0.291	**0.319**
RiverFlowsThroughCity	0.000	0.001	0.052	**0.076**
SportsTeamPositionForSport	0.217	**0.217**	0.178	0.180
StadiumLocatedInCity	0.090	0.156	0.275	**0.321**
StateHasLake	0.000	0.000	**0.000**	0.000
TeamPlaysInLeague	0.934	0.936	**0.947**	0.939
WriterWroteBook	0.144	0.179	0.195	**0.202**

Table 4: Average precision for each relation tested on the NELL KB. The best performing method on each relation is bolded.

Relation	KB	KB + SVO	KB + C-SVO	KB + V-SVO
/amusement_parks/park/rides	0.000	0.009	0.004	**0.013**
/architecture/architect/structures_designed	0.072	0.199	0.257	**0.376**
/astronomy/constellation/contains	0.004	**0.017**	0.000	0.008
/automotive/automotive_class/examples	0.003	0.001	0.002	**0.006**
/automotive/model/automotive_class	0.737	0.727	0.742	**0.768**
/aviation/airline/hubs	0.322	0.286	0.298	**0.336**
/book/literary_series/author_s	0.798	0.812	0.818	**0.830**
/computer/software_genre/software_in_genre	0.000	0.001	0.001	**0.001**
/education/field_of_study/journals_in_this_discipline	0.001	**0.003**	0.003	0.001
/film/film/rating	**0.914**	0.905	0.914	0.905
/geography/island/body_of_water	0.569	0.556	0.580	**0.602**
/geography/lake/basin_countries	0.420	0.361	0.409	**0.437**
/geography/lake/cities	0.111	0.134	**0.177**	0.175
/geography/river/cities	0.030	0.038	0.045	**0.066**
/ice_hockey/hockey_player/hockey_position	0.307	0.243	0.222	**0.364**
/location/administrative_division/country	0.989	0.988	**0.991**	0.989
/medicine/disease/symptoms	0.061	**0.078**	0.068	0.067
/medicine/drug/drug_class	**0.169**	0.164	0.135	0.157
/people/ethnicity/languages_spoken	0.134	**0.226**	0.188	0.223
/spaceflight/astronaut/missions	0.010	0.186	0.796	**0.848**
/transportation/bridge/body_of_water_spanned	0.534	0.615	0.681	**0.727**
/tv/tv_program_creator/programs_created	0.164	0.179	0.163	**0.181**
/visual_art/art_period_movement/associated_artists	0.044	0.040	**0.046**	0.037
/visual_art/visual_artist/associated_periods_or_movements	0.276	**0.295**	0.282	0.290

Table 5: Average precision for each relation tested on the Freebase KB. The best performing method on each relation is bolded. For space considerations, “Clustered SVO” is shortened to “C-SVO” and “Vector SVO” is shortened to “V-SVO” in the table header.

Vector SVO) on Freebase, (where 21% of queries gave no prediction). Our methods thus also improve MAP when calculated in this manner, but it is not an entirely fair metric,[5] so we use standard MAP to present our main results.

One interesting phenomenon to note is a novel use of the ALIAS relation in some of the relation models. The best example of this was found with the relation `/people/ethnicity/languages_spoken`. A high-weighted feature when adding surface relations was the edge sequence <ALIAS, ALIAS INVERSE>. This edge sequence reflects the fact that languages frequently share a name with the group of people that speaks them (e.g., Maori, French). And because PRA can generate compositional features, we also find the following edge sequence for the same relation: <`/people/ethnicity/included_in_group`, ALIAS, ALIAS INVERSE>. This feature captures the same notion that languages get their names from groups of people, but applies it to subgroups within an ethnicity. These features would be very difficult, perhaps impossible, to include in systems that do not distinguish between noun phrases and knowledge base entities, such as the graphs constructed by Gardner et al. (2013), or typical relation extraction systems, which generally only work with noun phrases after performing a heuristic entity linking.

7 Conclusion

We have offered two main contributions to the task of knowledge base inference. First, we have presented a new technique for combining knowledge base relations and surface text into a single graph representation that is much more compact than graphs used in prior work. This allowed us to apply methods introduced previously to much larger problems, running inference on a single machine over the entire Freebase KB combined with tens of millions of surface relations. Second, we have described how to incorporate vector space similarity into random walk inference over knowledge bases, reducing the feature sparsity inherent in using surface text. This allows us to combine distributional similarity with symbolic logical inference in novel and effective ways. With experiments on many relations from two separate knowledge bases, we have shown that our methods significantly outperform prior work on knowledge base inference.

The code and data used in the experiments in this paper are available at http://rtw.ml.cmu.edu/emnlp2014_vector_space_pra/.

Acknowledgments

This research has been supported in part by DARPA under contract number FA8750-13-2-0005, by NSF under grant 31043,18,1121946, and by generous support from Yahoo! and Google.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*.

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *Proceedings of EMNLP*. Association for Computational Linguistics.

Aapo Kyrola, Guy Blelloch, and Carlos Guestrin. 2012. Graphchi: Large-scale graph computation on just a pc. In *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 31–46.

Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67.

Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of EMNLP-CoNLL*.

Douglas B Lenat. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38.

Hugo Liu and Push Singh. 2004. Conceptnet: a practical commonsense reasoning tool-kit. *BT Technology Journal*, 22(4):211–226.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534. Association for Computational Linguistics.

[5] MAP is intended to include some sense of recall, but excluding queries with no predictions removes that and opens the metric to opportunistic behavior.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit.

Pablo N. Mendes, Max Jakob, and Christian Bizer. 2012. Dbpedia for nlp: A multilingual cross-domain knowledge base. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. *arXiv preprint arXiv:1404.5367*.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of NAACL-HLT*.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013a. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of WWW*.

Partha Pratim Talukdar, Derry Wijaya, and Tom Mitchell. 2012. Acquiring temporal constraints between relations. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 992–1001. ACM.

William Yang Wang, Kathryn Mazaitis, and William W. Cohen. 2013. Programming with personalized pagerank: A locally groundable first-order probabilistic logic. In *Proceedings of the 22Nd ACM International Conference on Conference on Information &#38; Knowledge Management*, CIKM '13, pages 2129–2138, New York, NY, USA. ACM.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *WWW*.