Observed versus latent features for knowledge base and text inference

Kristina Toutanova
Microsoft Research
Redmond, WA, USA

Danqi Chen*
Computer Science Department
Stanford University

Abstract

In this paper we show the surprising effectiveness of a simple observed features model in comparison to latent feature models on two benchmark knowledge base completion datasets, FB15K and WN18. We also compare latent and observed feature models on a more challenging dataset derived from FB15K, and additionally coupled with textual mentions from a web-scale corpus. We show that the observed features model is most effective at capturing the information present for entity pairs with textual relations, and a combination of the two combines the strengths of both model types.

1 Introduction

Representing information about real-world entities and their relations in structured knowledge bases (KBs) enables numerous applications. Large, collaboratively created knowledge bases have become recently available (some examples are Freebase (Bollacker et al., 2008), YAGO (Suchanek et al., 2007), and DBPedia (Auer et al., 2007)), but even though they are impressively large, their coverage is far from complete. This has motivated research in automatically deriving new facts to extend a manually built knowledge base, by using information from the existing knowledge base and information from textual mentions of entities in documents.

Many statistical models for predicting new links in knowledge bases have been applied to this task, with most successful ones being latent feature models that learn continuous representations for entities and relations (Bordes et al., 2011; Nickel et al., 2011; Bordes et al., 2013), and observed feature models which predict based on observable features in the knowledge graphs (Lao et al., 2011; Riedel et al., 2013). Additionally, studies have looked at the contribution of text-based extraction to knowledge base completion (Lao et al., 2012; Gardner et al., 2013).

In this paper we compare empirically a very simple observed features model to state-of-the-art latent feature models recently applied to two commonly used datasets for knowledge base completion: a dataset adapted from the Freebase KB, called FB15K (Bordes et al., 2013) and a dataset derived from the WordNet graph WN18, also introduced in (Bordes et al., 2013). We show that the simple observed features model substantially outperforms latent feature models, possibly due to the arguably unrealistic redundancy in the KB graphs of these datasets. Nevertheless, it is intriguing that the latent feature models studied are not able to learn the target concept as well, even given a large number of latent features.

We also construct a harder, perhaps more realistic dataset derived from FB15K, in which we remove near-duplicate or inverse-duplicate relations. We show that in this new dataset our studied latent feature models substantially outperform the observed feature models. When we augment the newly constructed dataset with textual mentions derived from the ClueWeb 12 web-scale document collection, we see that the observed features model is more powerful than the latent feature models, but also that a combination of the two is superior to either of them.

2 Related Work

There has been a large amount of work on statistical models for knowledge base completion. Nickel et al. (2015) provide a recent overview.

Most related to our current focus is recent work applying latent feature models to the FB15K and WN18 datasets (Bordes et al., 2013; Wang et al., 2014; Yang et al., 2015), work containing comparisons between observed and latent feature mod-

* Parts of this research were conducted during the author's internship at Microsoft Research.

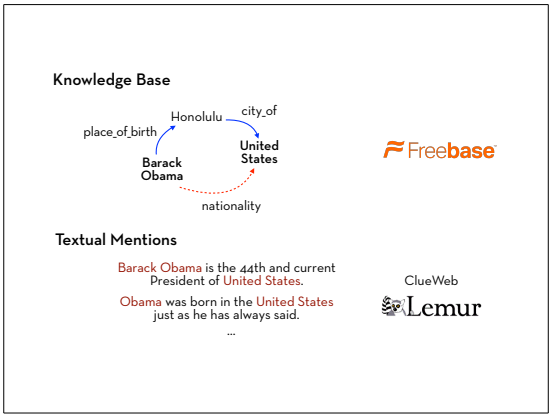

Figure 1: A knowledge base fragment coupled with textual mentions of pairs of entities.

els (Dong et al., 2014; Nickel et al., 2014), and work using inference from both text and knowledge base relations (Lao et al., 2012; Riedel et al., 2013; Dong et al., 2014; Gardner et al., 2014).

Our work differs from this prior work in that we compare a very simple form of observed feature models, based on using only direct links between candidate entity pairs, to state-of-the-art latent feature models on two benchmark datasets, with surprising results.

Our work on using textual mentions for knowledge base inference differs from prior work in the scale and richness of the knowledge base and textual relations used, as well as in that we evaluate the impact of text not only on mentioned entity pairs like (Gardner et al., 2014; Riedel et al., 2013) but on all links. We represent knowledge base and textual patterns in a single knowledge graph, like Lao et al. (2012) and Riedel et al. (2013), but refine the learning method to treat textual relations differently in the loss function, to maximize predictive performance on the knowledge base relations. We show the impact of observed and latent feature models and their combination in knowledge graphs with and without textual relations.

3 Models for knowledge base completion

We begin by introducing notation to define the task, largely following the terminology in Nickel et al. (2015). We assume knowledge bases are represented using RDF triples, in the form (*subject*, *predicate*, *object*), where the subject and object are entities and the predicate is the type of relation. For example, the KB fragment shown in Figure 1 is shown as a knowledge graph, where the entities are the nodes, and the relations are shown as directed labeled edges: we see three entities which participate in three relation instances indicated by the edges.

The task we are interested in is, given a training KB consisting of entities with some relations between them, to predict new relations (links) that do not appear in the training KB. For example, the triple (*Barack Obama*, *nationality*, *United States*) could be predicted from the training KB triples (*Barack Obama*, *place_of_birth*, *Honolulu*) and (*Honolulu*, *city_of*, *United States*). More specifically, we will build models that rank candidate entities for given queries $(e_1, r, ?)$ or $(?, r, e_2)$, which ask about the subject or object of a given relation.

The following notation will help us define the statistical models over knowledge graphs that we consider. Let $\mathcal{E} = (e_1, e_2, \ldots, e_{N_e})$ denote the set of entities in the knowledge graph and let $\mathcal{R} = (r_1, r_2, \ldots, r_{N_r})$ denote the set of relation types. We denote each possible triple as $x_{i,j,k} = (e_i, r_k, e_j)$ and model its presence with a binary random variable $y_{i,j,k} \in \{0, 1\}$ which indicates whether the triple exists. We will focus on models that score possible triples $x_{i,j,k}$ using either *observed features* from the knowledge graph or *latent features* of the three elements of the triple. Both model classes use scoring functions $f(x_{i,j,k}; \Theta)$ that represent the model's confidence in the existence of the triple. We first specify the forms of scoring functions we consider in this study, and later detail the loss functions used for training model parameters. We use the same loss function (as a function of triple scores) for training all models in this study.

3.1 Observed feature models

We consider an extremely simple form of observed feature models, which can be seen as an impoverished variant of path ranking (PRA) for KB completion (Lao and Cohen, 2010; Lao et al., 2011). In particular we define features for existing paths of length one for candidate triples (e_i, r_k, e_j). These can be paths from e_i to e_j or from e_j to e_i. Length one paths from e_i to e_j: we define binary features of the form $\mathbf{1}(r' \& r_k)$, which fire when the triple e_i, r', e_j exists in the training knowledge graph, and $r' \neq r_k$. This feature type captures correlations among multiple relation types for the same entity pair – for example, if someone lives in a certain city, they might be likely to work in the same city. Length one

paths from e_j to e_i: we define binary indicator features of the form $\mathbf{1}(r'_{inv}\&r_k)$, which fire when the triple e_j, r', e_i exists in the training knowledge graph. Here r' can capture the correlation with inverse relations, for example *nationality* and *people_of_nationality*.

Such features will fire only if the candidate entity pair (e_i, e_j) is already directly connected in the training knowledge graph (by a link in either direction). Thus such features are expected to be helpful only when there are multiple correlated relation types that tend to connect similar sets of entity pairs. In the experiments section, we will show that this is indeed the case for two commonly used KB completion datasets we study. It is also true for knowledge graphs augmented with textual links, where each co-occurrence of (e_i, e_j) in a document collection induces a link of a textually-defined relation type. In addition to the features looking at length one paths, for the observed feature models we define an indicator feature for every entity and relation in the triple. This captures a bias for these entities to occur in the subject or object position of the relation. The features are $\mathbf{1}(e_i = s\&r_k)$ and $\mathbf{1}(e_j = o\&r_k)$, where s and o indicate the subject and object positions, respectively. These features can capture the frequency with which each argument of the relation is occupied by a specific entity. For example, we can learn that *United States* is a common nationality for entities in Freebase.

Given a feature vector $\Phi_{i,j,k}$, the score of a triple is defined by its dot product with a parameter vector, which contains a weight for each feature: $f(x_{i,j,k}; \Theta) = \Phi_{i,j,k}^T \Theta$.

3.2 Latent feature models

In latent feature models, the score of a candidate triple is assumed to depend only on learned latent features of the entities and relations, and possibly additional global parameters. In this work we consider two simple latent feature models, which have been found to be competitive or outperform more complex alternatives in prior work (Yang et al., 2015; Riedel et al., 2013).

The first model we consider is model E (abbreviated from ENTITY), which captures the compatibility between entities and the subject and object positions of relations. It can be seen as learning a soft notion of entity types. The model was applied to knowledge-base completion for text-augmented knowledge graphs using a universal schema approach (Riedel et al., 2013). For each relation type, the model learns two latent feature vectors r_s and r_o of some dimensionality K. For each entity (node) e_i, the model also learns a latent feature vector n_i of the same dimensionality. The model is depicted in Figure 2. The score of a candidate triple (e_s, r, e_o) where the sub-scripts s and o are used to indicate subject and object positions, respectively, is defined as: $f(x_{s,r,o}) = {r_s}^T n_s + {r_o}^T n_o$.

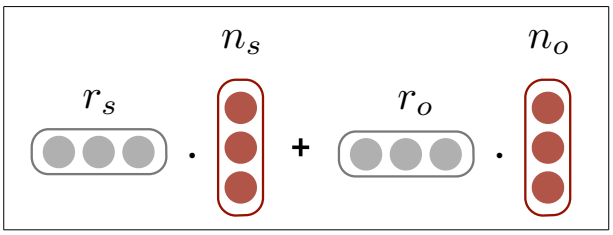

Figure 2: The continuous representations for model E.

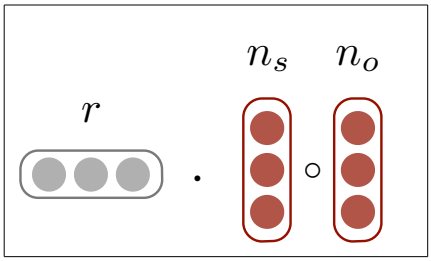

Figure 3: The continuous representations for model DISTMULT.

The second model, DISTMULT, is a special form of a bilinear model like RESCAL (Nickel et al., 2011), where the non-diagonal entries in the relation matrices are assumed to be zero. This model was proposed in Yang et al. (2015) under the name DISTMULT, and was shown to outperform the more highly parameterized bilinear model, as well as the additive model TRANSE (Bordes et al., 2013). In this model, each entity e_i is assigned a latent feature vector (embedding) n_i of dimensionality K and each relation type is assigned an embedding r of the same dimensionality. The model form is shown in Figure 3. The score of a candidate triple (e_s, r, e_o) is defined as: $f(x_{s,r,o}) = r^T(n_s \circ n_o)$.

If there are N_e entities, N_r relations, and latent feature vectors of dimensionality K are used, model E has $KN_e + 2KN_r$ parameters and model DISTMULT has $KN_e + KN_r$ parameters.

Combined models

We also consider weighted combinations of la-

tent feature models and observed feature models in a method similar to the one used in the Additive Relational Effects Model of Nickel et al. (2014). Given scoring functions $f_1(x_{i,j,j}, \Theta_1)$ and $f_2(x_{i,j,j}, \Theta_2)$ defined by two different models, we define a combined model for which the score of a triple is a weighted combination of the scores by the two models $w_1 f_1(x_{i,j,j}, \Theta_1) + w_2 f_2(x_{i,j,j}, \Theta_2)$. The component models could be latent of observed feature models, and the combination weights are either uniform (set to 1), or non-uniform and selected via a grid search on a validation set. We train the parameters of combined models jointly, by minimizing the loss function based on the combined scores.

3.3 Training loss function

Our loss function is motivated by the link prediction task and the performance measures used to assess model performance. As mentioned earlier, the task is to predict the subject or object entity for given held-out triples (e_1, r, e_2), i.e. to rank all entities with respect to their likelihood of filling the respective position in the triple. We would thus like the model to score correct triples (e_1, r, e_2) higher than incorrect triples (e', r, e_2) and (e_1, r, e') which differ from the correct triple by one entity. One could use a margin-based loss function as used in several approaches (Nickel et al., 2015). We use an approximation to the negative log-likelihood of the correct entity filler. We define the conditional probabilities $p(e_2|e_1, r)$ and $p(e_1|r, e_2)$ for object and subject entities given the relation and the other argument as follows: $p(e_2|e_1, r_k; \Theta) = \frac{e^{f(x_{e_1,e_2,k};\Theta)}}{\sum_{e'_2 \in Neg(e_1,r_k,?)} e^{f(x_{e_1,e'_2,k};\Theta)}}$.

Here the denominator is defined using a set of entities that do not fill the object position in any relation triple $(e_1, r, ?)$ in the training knowledge graph. Since the number of such entities is impractically large, we sample negative triples from the full set (we use 200 negative examples in our experiments). In some settings we also limit the candidate entities to ones that have types consistent with the position in the relation triple (Chang et al., 2014; Yang et al., 2015). We derive approximate type information automatically (as discussed below), but such information could also be present in the knowledge graphs.

Conditional probabilities for subject entities given relation and object are defined analogously, as follows: $p(e_1|r_k, e_2; \Theta) = \frac{e^{f(x_{e_1,e_2,k};\Theta)}}{\sum_{e'_1 \in Neg(?,r_k,e_2)} e^{f(x_{e'_1,e_2,k};\Theta)}}$

Given the definition of subject and object conditional probabilities for triples, our training loss function is defined as the sum of the negative log-probabilities of observed triples, also including an L2 penalty on the model parameters. If X denotes the set of all triples in the training knowledge graph, the training loss is defined as:

$$\begin{aligned} L(X, \Theta, \lambda) \quad = \quad & - \sum_{x_{e_1,e_2,r_k} \in X} \log p(e_2|e_1, r_k; \Theta) \\ & - \sum_{x_{e_1,e_2,r_k} \in X} \log p(e_1|r_k, e_2; \Theta) \\ & + \lambda \Theta^T \Theta \end{aligned}$$

3.3.1 Entity types

We define the type of an entity e as a pair of sets of relation types [R^s,R^o]; R^s is the set of relation types r for which e is the source node of a link with type r in the training knowledge graph and R^o is the set of relation types for which e is the target node of link with type r. For each relation, we compute a set of allowable entity types by checking the percentage of its arguments that have a given type and restricting the allowable types to the top t (chosen on a validation set, usually two or three). For example, for the subject position of a *parents* relation the most frequent type would be $parent^s$ (meaning subject of *parent*). A second frequent type might be $born\_in^s$ meaning subject of *born_in*. Using this construction we define the compatibility between entities and relation argument positions, which prunes the space of candidates quite significantly in many cases, while still maintaining a high upper bound on achievable performance. Details on the impact of usage of types are presented in Section 4.

3.4 Representation and loss for text-augmented knowledge graphs

In addition to knowledge graphs containing only relations r from a given manually developed ontology, we consider knowledge graphs augmented with textual relations derived from sentence co-occurrences of entity pairs. This follows the approaches of Lao et al. (2012) anf Riedel et al. (2013), who represent both textual and knowledge base relations in a single graph of "universal" relations, which allows joint reasoning from

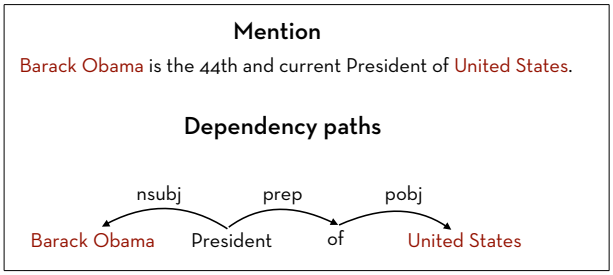

Figure 4: Textual relation extracted from an entity pair mention.

the two types of relations. Figure 4 shows the lexicalized dependency path between two entities that occur together in a sentence. An instance of textual relation of type "$\xleftarrow{\text{nsubj}}$ *president* $\xrightarrow{\text{prep}}$ *of* $\xrightarrow{\text{obj}}$ *United States*", corresponding to the sentence is added to the knowledge graph based on this occurrence. Textual co-occurrences of entity pairs often express relations between the entities, which might correspond exactly or approximately to knowledge base relations. Text could thus be a strong signal for predicting knowledge base relations (Lao et al., 2012).

Once a knowledge graph is augmented with textual relations, we can train the same models as before, treating knowledge base and text relations in a uniform manner. However, since we are only interested in predicting knowledge-base relations, it might be suboptimal for the model to try to fit its parameters toward predicting textual relations as hard as it tries to optimize for knowledge base relations. In other words, the part of the loss function looking at subject and object probabilities for textual relations t is only useful to provide an auxiliary prediction task which is beneficial for the main task using a multi-task learning setting. Therefore, one might choose an optimal weight τ which could be expected to be less than the weight of the primary loss function. We thus consider a modified loss function for KB+text model training, defined as follows. If the set of all triples using KB relations is X and the set of all triples using text relations is T, the loss is defined as $L(X \cup T, \Theta) = L(X, \Theta) + \tau L(T, \Theta)$. In the experiments section we will see that this simple modification can provide large performance benefits.

4 Experiments

We perform experiments with latent and observed feature models and their combination. We first present results on the FB15K dataset, which was originality constructed (using Freebase) by Bordes et al. (2013) and was subsequently used in several research studies (Wang et al., 2014; Yang et al., 2015). The number of relations and triples in the training, development and test portions of the datasets are given in Table 5.

4.1 Task and Evaluation Protocol

Given a set of triples in a set disjoint from a training knowledge graph, we test models on predicting the subject or object of each triple, given the relation type and the other argument. We rank all entities in the training knowledge base in order of their likelihood of filling the argument position. We report the mean reciprocal rank of the correct entity, as well as HITS@10 – the percent of test triples for which the correct argument was ranked in the top ten. We use *filtered* measures following the protocol proposed in Bordes et al. (2013) – that is, when we rank entities for a given position, we remove all other entities that are known to be part of an existing triple in the training, development, or test set. This avoids penalizing the model for ranking other correct fillers higher than the tested argument. We thus report filtered mean reciprocal rank (labeled MRR in the Figures), and filtered HITS@10. In the figures we present MRR values scaled by 100, so that the maximum possible MRR is 100.

Implementation details and hyper-parameter settings

For all models implemented in this work, we trained the models using the loss function presented in Section 3.3, using $\lambda = 1$ as the weight of the $L2$ regularizer. We used a batch learning parameter optimization method, after initial experiments showed it did better than stochastic optimization using AdaGrad. We experimented with LBFGS (Liu and Nocedal, 1989) and RProp (Riedmiller and Braun, 1993), and found RProp to converge faster to similar values of the objective for the latent feature models. All reported results use RProp. We also used early stopping to terminate optimization when the MRR on the validation set stopped improving.

We chose the optimal number of latent features via a grid search to optimize MRR on the validation set, testing the values 10,50,100,200,500,and 1,000. Similarly, we performed a grid search over the values of the parameter τ which weighs

Dataset	Relations	Entities	Triples in Train / Validation / Test			% Test Linked
FB15K	1,345	14,951	483,142	50,000	59, 071	80.9
FB15KSelected	237	14,541	272,115	17,535	20, 466	0
WN18	18	40,943	141,442	5,000	5,000	94.0

Figure 5: Datasets used in this study. FB15K and WN18 have been used in prior work and FB15KSelected is a new dataset derived from FB15K and ClueWeb (described in text).

Model	FB15K Type Constraints		FB15K No Type Constraints	
	MRR	HITS@10	MRR	HITS@10
E	22.7	34.0	21.8	33.6
DISTMULT	63.1	79.0	55.5	79.7
E+DISTMULT	65.9	81.0	56.2	78.3
TransE			32.0	53.9
DISTMULT (Yang et al., 2015)			36.0	57.7
TransH (bern.) (Wang et al., 2014)				64.4
NodeFeat	21.7	32.6	21.6	32.4
LinkFeat	79.1	80.8	77.9	80.4
Node+LinkFeat	**82.1**	**86.1**	82.2	87.0

Figure 6: Results on FB15K with and without type constraints for candidate filtering in training and testing.

the textual relations loss, testing values in the set $\{0.01, 0.1, 0.25, 0.5, 1\}$.

4.2 Experiments on KB Completion using FB15K and WN18

We present experiments of different models introduced in this paper on these datasets, and additionally include results reported in prior work. We also evaluate the impact of our use of types as hard constraints in training and testing, and how these constraints impact latent feature models versus observed feature models.

Figure 6 presents the results under two settings: using automatically derived types versus not using them. The results not using types are presented in the right half of the Figure. The first six rows report performance measures obtained using latent feature models. The first three models presented are the ones defined in Section 3.2 and implemented in this work. We evaluate these models when type constraints are used or when they are not used. The next three rows report results from prior work by directly copying reported numbers from the respective papers. Since these papers did not use type constraints, we list the results in the right two columns only. The model TransE was proposed in (Bordes et al., 2013) but we use the results from the implementation of (Yang et al., 2015), because these results were higher. The TransH (bern.) results are obtained by the model presented in (Wang et al., 2014).

The last three rows show results from observed feature models, as defined in Section 3.1, where the first model uses only node features, the second uses only direct link features, and the third uses both feature types.

The type constraints were defined using the method presented in Section 3.3.1. We choose the best settings for the method based on coverage of the correct triples in the validation set. Given the hard pruning of candidates by type filtering, the method using types has less than 100 percent achievable accuracy — the oracle HITS@10 by using type constraints is 98.3. We found that the number of latent features did not have a large impact on performance for model E, but did have large impact for the other two models. Using 500 hidden dimensions was optimal for these two models. Even though the form of the scoring function for DISTMULT is exactly the same as defined in (Yang et al., 2015), we obtain much higher performance. We attribute this to the larger number of hidden dimensions (500 vs 100), and the use of the softmax-based loss function with 200 negative examples and batch training.[1] As seen, the impact of the type constraints is large and positive, especially on the MRR values. Our implementation of these embedding models outperforms the other recent results by TransH (Wang et al., 2014), which we also attribute to the loss function and optimization.

The most striking result on this dataset is seen in the last two rows of the Table, where we can

[1] We chose the optimal number of hidden dimensions to optimize MRR on the development set. We found that performance improved with dimensionality up to 500 dimensions. For DISTMULT with type constraints, the MRR for dimensionality $\{10, 100, 500\}$ was $\{35.2, 55.8, 63.1\}$, respectively.

Model	FB15KSelected					
	MRR a/t/nt			HITS@10 a/t/nt		
	Without Text					
E	23.5	20.2	24.4	35.6	31.7	36.9
DISTMULT	25.3	20.9	26.7	40.8	34.8	42.6
E+DISTMULT	26.6	23.1	27.7	43.0	38.4	44.4
NodeFeat	23.5	20.3	24.5	35.6	31.7	36.8
LinkFeat	6.3	3.1	7.3	7.9	5.2	8.7
Node+LinkFeat	22.6	19.3	23.7	34.7	30.6	36.0
Combined	26.8	23.1	28.0	42.8	37.8	44.3
	With Text					
E+DISTMULT ($\tau = 1$)	26.6	24.0	27.3	41.3	38.4	42.2
E+DISTMULT ($\tau = .1$)	27.4	24.3	**28.3**	43.8	39.8	**45.0**
Node+LinkFeat	27.2	**39.6**	23.4	41.4	**60.5**	35.5
Combined (($\tau = .1$)	**29.3**	**39.1**	26.3	**46.2**	**60.0**	41.9

Figure 7: Results on FB15KSelected with and without addition of textual links. Model Combined is a combination of the latent feature models E and DISTMULT and the observed feature model Node+LinkFeat.

Model	WN18	
	MRR	HITS@10
TRANSE (Bordes et al., 2013)		89.2
DISTMULT (Yang et al., 2015)	83.0	94.2
BILNEAR (Yang et al., 2015)	89.0	92.8
TransH (unif) (Wang et al., 2014)		86.7
NodeFeat	2.9	5.0
LinkFeat	93.8	93.9
Node+LinkFeat	94.0	94.3

Figure 8: Results on WN18 using performance reported in prior work as well as performance of observed feature models.

see the performance of the observed feature models based on direct links. The performance of these models (in MRR) is much higher that the performance of the latent feature models obtained in this work and in prior work. This is perhaps not so surprising when we look at the number of test set triples (e_1, r, e_2) for which either (e_1, r', e_2) or (e_2, r', e_1) occur in the training set – i.e., which are directly linked in the training knowledge graph. This number is almost 81% (reported in Table 5), and explains why the observed features model which uses this information directly can do so well. What is more surprising is that latent feature models have not approached this performance, even given a large number of latent feature dimensions. We see this is an interesting datapoint which can motivate analysis and improvement in the state-of-the-art in knowledge base completion using latent variable models.

Two other interesting results from these experiments are that the observed feature model using only entity features (NodeFeat) has almost the same performance as the latent feature model E and both can be seen as learning a unigram distribution over entities for argument positions of relations. Additionally, the observed feature models are not substantially affected by the use of type constraints, since they effectively learn to model similar type concepts using the features.

We also tested the models on WN18, and report results from prior work using latent feature models as well as our implementation of the observed feature models in Figure 8. As seen, the observed feature models using link features strongly outperform prior work in the MRR measure (achieving around 45% error reduction over the best previously reported results), and are comparable to the best models according to the HITS@10 measure. As shown in Table 5, 94.0 of test triple entities are directly linked in the training KB, explaining the success of these simple models.

Given our analysis of the FB15k and WN18 datasets and the power of a simple observed features model, we are motivated to construct a more realistic knowledge base completion dataset for which we can assume that trivially entailed facts (due to relation symmetry or the presence of inverse relations) have already been inferred and the task is to entail facts requiring non-trivial inference. To this effect we construct a subset of FB15K, which we term FB15KSelected, and which represents a more challenging learning setting.

4.3 Experiments using knowledge graph and text inference on FB15KSelected

The dataset FB15KSelected[2] was constructed by first limiting the set of relations in FB15K to the most frequently used 401 relations (a setting using this subset of frequent relations was also used in (Yang et al., 2015)). We then automatically detected near-duplicate and inverse relations by checking whether the set of entity pairs in the relations is either almost the same (at least 97% of the pairs are in the intersection), or whether the set of inverse entity pairs is almost the same e.g. comparing $[e_1, e_2]$ for r to the set $[e_2, e_1]$ for r'. For example, this process detected that the relation */award/award_nominee* is inverse of */award_nominee/award*. Given this information, we filtered the set of relations to keep only one of a set of inverse or duplicate relations; this resulted in 237 relations, and we limited the training, validation, and development set triples to these relations. We also filtered from the validation and test sets any triples whose entity pairs were directly linked in the training database. Such direct links could admittedly be legitimately present in a realistic scenario but we excluded them to avoid additional trivial cases which could have not been detected via the prior filtering step. The statistics for this resulting dataset are shown in Table 5.

While for this more realistic dataset we have excluded all direct KB links for test entity pairs, there is a realistic source of direct relations between test entity pairs – textual relations expressed by sentences containing these pairs of entities. We use the ClueWeb12 [3] corpus coupled with Freebase mention annotations (Gabrilovich et al., 2013) to extract textual relations for all entity pairs in the knowledge base. We extract textual patterns from 200 million dependency-parsed sentences and we represent the textual relations via the fully lexicalized dependency path connecting two entities, as shown in Figure 4. After pruning, we use 25,000 unique textual relations and add links to the training knowledge graph based on these relations. There are 6.6 million links induced from the textual relations for the FB15KSelected knowledge base. Of the test KB triples, 23.3% of the entity pairs have textual mentions. For the training set, 31% of the entity pairs that have a KB link have a textual mention, and having a mention increases the chance that a random entity pair has a relation from .1% to 4.2% – a forty-fold increase.

Figure 7 shows the results for this dataset – the upper portion contains results for models not using textual mentions, and the lower part contains results of models also using the text. The results are shown using the MRR and HITS@10 measures, and these are further broken down into overall/with textual mentions/without textual mentions (a/t/nt).

For the setting where no textual mentions are used, we see that the latent feature models outperform the observed feature models (since there are no direct links in the training set for test triples, the observed features model LinkFeat has performance which is random subject to the type constraints (and where ties are broken by order of appearance in the training set)). Using node features only is best for the observed feature models, and the overall MRR of this model (23.5), is substantially below that of the best latent feature model, E+DISTMULT with overall MRR of 26.6. A model which combines the latent and observed features (shown in row 7), does not bring substantial improvement.

The second (lower) part of the Figure shows model results when the training knowledge graph is expanded with textual relations. First, for the best latent feature model E+DISTMULT which treats knowledge base and textual relations uniformly, using $\tau = 1$ as in the universal schema approach (Riedel et al., 2013), we see no improvement from using the textual mentions. Indeed, there is an improvement in MRR on the test triples that have mentions (23.1 to 24.0), but performance degrades on the more numerous test cases without mentions. When τ is optimized via grid search to a value of $\tau = .1$ we see a good improvement to overall MRR 27.4 due to using text, which holds for cases with mentions as well as ones without mentions. The observed features model benefits from text strongly, and in particular the MRR on test triples with mentions increases from 19.3 to 39.6. The performance on triples without mentions is very low, however. Since the latent and observed feature models have complementary strengths, their combination (last row in the Figure) substantially outperforms both kinds of models, reaching an overall MRR of 29.3 and overall

[2] A release of the dataset will be available soon. Contact the authors for more details if interested.

[3] http://www.lemurproject.org/clueweb12.php/

HITS@10 of 46.2. The MRR on test triples with mentions is almost doubled compared to the models not using text.

Conclusion

This work provided two main lessons for knowledge base completion. First, we showed that the presence of relations between candidate pairs can be an extremely strong signal in some cases, and that this signal was not effectively captured by the studied latent feature models. Second, we showed that textual links extracted from a large document collection and added to an existing KB-completion dataset brought substantial improvements, especially on test cases with textual occurrences. It was beneficial to use the direct textual links as features in an observed features model and to combine that with a latent feature model, to effectively capture inferences among KB relations and direct cues from text. We also showed that in a dataset where training and test triples are not artificially limited to only ones that have textual mentions, it becomes important to tradeoff the weight of the loss incurred from textual versus KB relations.

Acknowledgements

We would like to thank Jianfeng Gao, Scott Yih, Patrick Pantel, Michael Gamon, Hoifung Poon and the anonymous reviewers for useful suggestions.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. *Dbpedia: A nucleus for a web of open data*. Springer.

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 the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*, number EPFL-CONF-192344.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NIPS)*.

Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *International Conference on Knowledge Discovery and Data Mining (KDD)*.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: Freebase annotation of ClueWeb corpora, Version 1 (release date 2013-06-26, format version 1, correction level 0).

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 *Empirical Methods in Natural Language Processing (EMNLP)*.

Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 809–816.

Maximilian Nickel, Xueyan Jiang, and Volker Tresp. 2014. Reducing the rank in relational factorization models by including observable patterns. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1179–1187. Curran Associates, Inc.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A review of relational machine learning for knowledge graphs: From multi-relational link prediction to automated knowledge graph construction. *arXiv preprint arXiv:1503.00759*.

Sebastian Riedel, Limin Yao, Benjamin M. Marlin, and Andrew McCallum. 2013. Relation extraction with matrix factorization and universal schemas. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Martin Riedmiller and Heinrich Braun. 1993. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Neural Networks, 1993., IEEE International Conference on*, pages 586–591. IEEE.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations (ICLR)*.