A Hierarchical Model for Universal Schema Relation Extraction

Arvind Neelakantan, Alexandre Passos, Andrew McCallum
School of Computer Science
University of Massachusetts, Amherst
{arvind,apassos,mccallum}@cs.umass.edu

ABSTRACT

Relation extraction by *universal schema* avoids mapping to a brittle, incomplete traditional schema by instead making predictions in the *union* of all input schemas, including textual patterns. Modeling these predictions by matrix competition with matrix factorization has yielded state-of-the-art accuracies. One difficulty with prior work in matrix factorization, however, is that there is no negative training data. As a result, existing methods merely sample an entity-pair's unobserved relation types and assume they are negative. In this paper we instead maximize the likelihood of the observed data—achieving tractability by arranging the relation types as leaves in a binary tree. We show empirically that the choice of tree structure is consequential, and achieve a 2.61% F1 score improvement over the previous approach. Furthermore, a simple combination of this approach with the previous approach results in a 3.53% gain in F1 score.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing

Keywords

Universal Schema; Relation Extraction; Hierarchical Model

1. INTRODUCTION

Universal schema relation extraction proposed by Riedel et al. [17] considers the *union* of all involved schemas where relation types include textual surface patterns and relation types from a pre-existing database. This database is populated using a matrix factorization model that learns latent representations for entity pairs and relation types.

One of the main challenges in training universal schema models is that there are no observed negative examples. Riedel et al. [17] train the universal schema model using Bayesian Personalized Ranking (BPR) [15]. In BPR, for every positive observed example they randomly sample only one unobserved example and train with the objective of ranking the observed example above the unobserved example. This approach has at least two drawbacks. First, the randomly sampled unobserved data may actually correspond to a positive example. Second, the parameters are learned to establish an ordering (ranking) over only two examples. Due to the second drawback, the algorithm may require large number of training examples to estimate the latent vectors reliably.

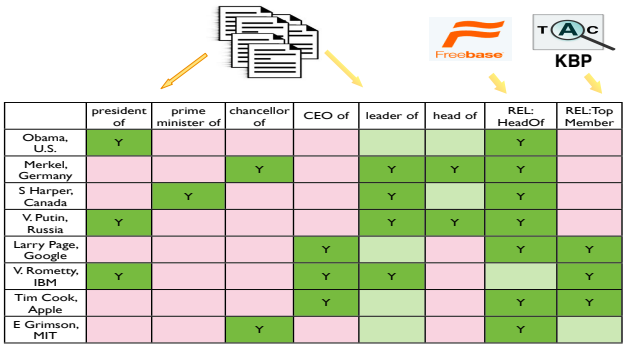
To avoid the drawbacks of this ranking-based approach, we instead maximize the likelihood of the observed data without making any such assumptions about the unobserved data. We do not assume that the observed positive examples must be ranked above unobserved examples. By maintaining a probability distribution over the set of relations for every entity pair, we avoid the need for explicit negative training data or randomly sampled "pseudo-negative" examples. The relation types are arranged as leaves in a binary tree for computational considerations. This method was proposed by Mnih et al. [10] as an alternative to BPR and performs better than BPR on a movie recommendation task.

We experiment with multiple methods for constructing a hierarchical structure over the set of relations and empirically measure the effect of the hierarchical structure on the final performance. Experimental results show that our approach achieves a 2.61% gain in F1 score over the previous approach described in Riedel et al. [17] and a simple combination of the two models results in a 3.53% gain in F1 score. We expect even higher gains if evaluated on a dataset that is constructed from a more complete knowledge base, which we are currently building.

2. UNIVERSAL SCHEMA RELATION EXTRACTION

Universal schema relation extraction considers the *union* of all involved schemas where relation types include textual surface patterns and relation types from a pre-existing database. More formally, let $R = \{r_1, r_2, \ldots, r_n\}$ be the set of relations and $E = \{(e_{1_1}, e_{1_2}), (e_{2_1}, e_{2_2}), \ldots, (e_{m_1}, e_{m_2})\}$ be the set of entity pairs that are observed. Note that the set R includes relation types that are textual patterns and relations taken from a database such as Freebase and TAC KBP schema. Binary decision variable $y_{i,j}$ is used to indicate whether the entity pair (e_{i_1}, e_{i_2}) $(i = 1, 2, \ldots, m)$ is in the relation r_j $(j = 1, 2, \ldots, n)$ or not. $y_{i,j}$ is 1 if the entity pair (e_{i_1}, e_{i_2}) is in the relation r_j and 0 otherwise.

In Riedel et al. [17], the entity pairs and the relation



	president of	prime minister of	chancellor of	CEO of	leader of	head of	REL: HeadOf	REL:Top Member
Obama, U.S.	Y						Y	
Merkel, Germany			Y		Y	Y	Y	
S Harper, Canada		Y			Y		Y	
V. Putin, Russia	Y				Y	Y	Y	
Larry Page, Google				Y			Y	Y
V. Rometty, IBM	Y			Y	Y			Y
Tim Cook, Apple				Y			Y	Y
E Grimson, MIT			Y				Y	

Figure 1: Entity Pairs and relation types are in the rows and the columns of the matrix respectively. Green "Y" are observed positive cells. Light green are predicted positive cells. Light red are predicted negative cells.

types are arranged in the rows and columns of a matrix respectively (Figure 1). The probability that an entity pair (e_{i_1}, e_{i_2}) is in relation r_j is modeled by,

$$P(y_{i,j} = 1 \mid \theta_{i,j}) = \frac{1}{\exp(-\theta_{i,j})}$$

where $\theta_{i,j} = \alpha_{i_1,i_2}.\gamma_j$, and $\alpha_{i_1,i_2} \in R^d$ is the latent representation for entity pair (e_{i_1}, e_{i_2}) and $\gamma_j \in R^d$ is the latent representation for relation r_j. The parameters of the model which are the latent representations of the entity pairs and the relation types are learned using matrix factorization. All training examples are positive examples where an entity pair is observed to be in a relation. These positive examples include observed textual relation instances and facts from a database. There are no observed negative training examples. Given a training example where $y_{i,j} = 1$, the parameters are estimated to rank $P(y_{i,j} = 1) > P(y_{k,j} = 1)$ where $y_{k,j}$ is a randomly sampled unobserved cell in the matrix [15].

The first obvious drawback of the training algorithm is that the randomly sampled unobserved example $y_{k,j}$ which is ranked below a positive observed example, may actually correspond to a positive example. The other drawback is that the parameters are learned by ranking only two data instances. So when a relation type has few positive training examples, only a small portion of the entity pairs are explicitly ranked lower than the positive examples. This can make the model predict this relation with a high score for many unobserved entity pairs because they are never explicitly ranked while training. We discuss this issue in more detail in Section 5. One impractical solution to this problem would be to explicitly rank the observed entity pair above all other entity pairs in the training data but that would make the method computationally intractable.

3. HIERARCHICAL MODEL

In this section, we describe a hierarchical model for universal schema relation extraction. In our approach, the likelihood of the observed data is maximized without making strict assumptions about the unobserved data. By maintaining a probability distribution over the set of relations for every entity pair, we avoid the need for negative training data and avoid randomly sampling unobserved data and speculatively treating them as negative examples.

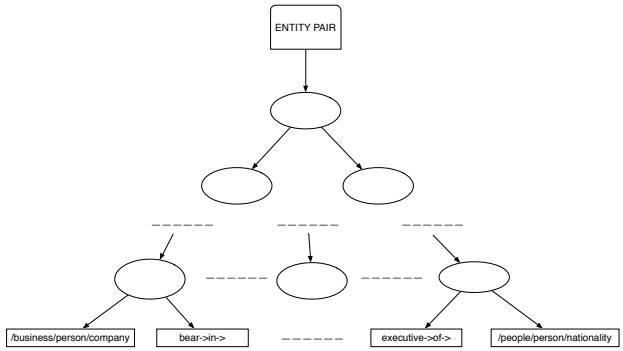


Figure 2: An example of a binary tree over the set of relations. Leaf nodes are relations and non-leaf nodes are binary logistic regression classifiers.

We first outline a simple but impractical approach to maximize the likelihood of the observed data. Here, we model the probability that an entity pair (e_{i_1}, e_{i_2}) is in relation r_j by,

$$P(y_{i,j} = 1 \mid \theta_{i,1}, \theta_{i,2}, \ldots, \theta_{i,n}) = \frac{\exp(\theta_{i,j})}{\sum_{k=1}^{n} \exp(\theta_{i,k})}$$

The model can be trained to maximize the likelihood of the training data using stochastic gradient descent. However, at each training iteration we would need to compute the normalization constant in the denominator which takes time linear in the number of relations. This limits the applicability of the method when there are a large number of relations. In universal schema, we consider both textual patterns and relation types from a pre-existing database, hence scalability is important.

To overcome the linear time complexity for each training iteration, we place the relations in a tree structure. The relations occupy exactly one leaf position in the binary tree (Figure 2). Each non-leaf node is a binary logistic regression classifier that computes the probability of going to the right child or the left child. Consider a non-leaf node ω whose input is $\alpha_{i_1,i_2} \in R^d$, the latent representation of the entity pair (e_{i_1}, e_{i_2}). The probability of moving from node ω to it's right child ω_r is given by,

$$P(\omega_r \mid \omega, \beta_\omega, \alpha_{i_1,i_2}) = \frac{1}{\exp(-\alpha_{i_1,i_2}.\beta_\omega)}$$

where β_ω is the parameter vector of the binary classifier at node ω. The probability of moving from node ω to it's left child ω_l is $P(\omega_l \mid \omega) = 1 - P(\omega_r \mid \omega)$.

Now, the probability that an entity pair (e_{i_1}, e_{i_2}) is in relation r_j is given by the product of the probabilities of the decisions that lead to the leaf node representing relation r_j from the root.

$$P(y_{i,j} = 1 \mid \alpha_{i_1,i_2}, \beta_{\omega_1}, \beta_{\omega_2}, \ldots, \beta_{\omega_j}) = \prod_{k=1}^{L_j} P(\omega_k^j \mid \omega_{k-1}^j, \alpha_{i_1,i_2}, \beta_{\omega_{k-1}})$$

where ω_0 is the root node and $\omega_0, \omega_1, \ldots, \omega_{L_{j-1}}$ are the sequence of nodes that lead to the relation node $j = \omega_{L_j}$ in the tree.

If we construct a balanced binary tree over the set of relations, the computational cost for a single training iteration is $O(log(n))$ since an observed example serves as a training example for $O(log(n))$ binary classifiers in the non-leaf

nodes. Hence, we can achieve exponential time speed up over the flat model by constructing a hierarchical structure over the set of relations.

The parameters in our model are the latent representations of the entity pairs and the weight vectors of the binary classifiers. We use FACTORIE [13] to implement our model. We maximize the likelihood using gradient descent with AdaGrad [3]. Training is done in parallel using a hogwild trainer [14].

3.1 Hierarchical Clustering for Learning Trees

We find that the structure of the tree over the set of relations plays a significant role in achieving good performance. We learn the tree structure by performing top-down hierarchical clustering on the vector representations of relations learned in Riedel et al. [17]. Starting from the root node, we partition the set of relations at each non-leaf node into two subsets (which go to the left and the right of the node in the tree) by applying k-means clustering to cluster the vector representations into two clusters.

We experiment with trees in which the relations are allowed to occupy multiple positions in the tree. This is achieved by assigning a relation to both the clusters in k-means if the vector representation of the relation is *close* to the boundary between the clusters. More formally, let the distance between the vector representation of a relation and the centroids of the first and second cluster be d_1 and d_2 respectively. If $(d_1 - d_2)/(d_1 + d_2) < \epsilon$ we allow the relation to be in both the clusters. Note that this can happen while running k-means at any node in the tree. As an extension of this technique, we fuse two trees, each of them obtained by running k-means with different random seeds. We fuse them by creating a new root node and attaching one tree to the left and the other tree to the right of the root node. We empirically study the effect of the hierarchical structure on the final performance in Section 5. Similar techniques have been explored to learn better trees over words to learn language models [8].

4. RELATED WORK

Relation Extraction: In some early work, a separate model is trained using supervised techniques for each target relation [2]. Distant supervision eliminates the need for supervised data by aligning sentences with a pre-existing database to induce labels for training supervised classifiers [7, 16, 1]. However, most available databases are incomplete by missing important relation types. Alternatively, OpenIE [4] considers textual surface patterns between mentions as relation types. A significant drawback of OpenIE is that it does not generalize across its many different relation types. To generalize better, methods like [5, 18] cluster textual forms but the learned clusters fail to capture asymmetric implicature. Universal schema relation extraction achieves state-of-the-art accuracy [17].

Hierarchical Model for Maximizing Likelihood: The use of tree structure to reduce normalization cost was introduced for language modeling [11]. Mnih et al. [10] develop a hierarchical model for collaborative filtering with *implicit feedback* in which there are no negative training examples as in our case. A similar training algorithm was also used to estimate word embeddings [6].

5. EXPERIMENTS

Model	Precision	Recall	F1 score
Huffman Tree	47.34	47.97	47.65
Learned Tree	58.24	61.16	59.66
Negative Sampling [17]	60.29	**62.27**	61.26
Learned Tree-1	61.55	62.04	61.80
Learned Tree-1 × 2	63.70	62.03	62.86
Simple Combination	**76.52**	54.15	**63.42**

Table 1: Precision, Recall, F1 scores measuring the effect of various tree structures. Results from Riedel et al. [17] are included for comparison.

In this section, we describe the data used in our experiments and discuss our experimental results.

5.1 Data

For our experiments, we use the data described in Riedel et al. [17] which has 208,226 entity pairs and 4476 relation types. Among the 4476 relation types, 4419 are textual surface patterns while the remaining 57 relations are taken from the Freebase schema. The textual relational instances are obtained by processing documents from NYTimes and we include all the observed textual surface relation instances during training. Textual mentions were entity linked to Freebase using a simple string matching heuristic. The Freebase facts are split equally into train and test facts, and the corresponding entity pairs into train and test entity pairs. We evaluate on Freebase facts that are hidden during training.

5.2 Results

Our predictions consist of instances in which the computed probability of the entity pair being in a relation is greater than a threshold which is tuned using development data. Table 2 shows results obtained using five different random balanced binary trees. The large variance indicates that the quality of the hierarchical structure has a significant impact on accuracy.

Table 1 shows the empirical effect of different hierarchical structures on the final performance. *Huffman Tree* refers to the tree obtained by performing Huffman encoding on the set of relations using counts from training data [6]. *Learned Tree* is the tree learned by top-down hierarchical clustering using representations from Riedel et al. [17]. *Learned Tree-1* refers to the tree in which relations occupy more than one leaf position. We set ϵ to 0.01 in our experiments. *Learned Tree-1 × 2* is obtained by fusing two trees, each of which is constructed by running k-means with different random seeds. *Negative Sampling* refers to the method described in Riedel et al. [17] and *Simple Combination* is a model which predicts an entity pair to be in a relation only if both *Negative Sampling* and *Learned Tree-1 × 2* predict it. The results indicate that the tree structure plays a crucial role in achieving good performance. Allowing relations to occupy multiple positions in the tree and fusing two trees helps accuracy. The best results are obtained by combining the predictions of the *Negative Sampling* and *Learned Tree-1 × 2* models.

5.3 Error Analysis

Here, we analyze the predictions made by the *Negative*

Relation	Training Examples	*Negative Sampling*		*Learned Tree-1 × 2*	
		TP	FP	TP	FP
/film/film/written_by	146	52	299	0	0
/film/film/produced_by	81	28	311	0	0
/sports/sports_team_owner/teams_owned	29	71	225	29	70
/people/person/place_lived	715	42	51	122	280
/business/person/company	979	572	38	773	160
/organization/parent/child	165	225	123	234	216

Table 3: True Positives (TP) and False Positives (FP) predicted by the two models.

Model	Precision	Recall	F1 score
Random-1	52.89	49.67	51.23
Random-2	50.28	47.48	48.73
Random-3	53.96	44.24	48.62
Random-4	54.28	44.94	49.17
Random-5	49.54	46.30	47.87

Table 2: Precision, Recall, F1 scores using five different random balanced trees showing huge variance in the results

Sampling model and the *Learned Tree-1 × 2* model. The true positives predicted by both these models are very similar. Both the models get nearly 4200 true positives out of which approximately 3700 are common predictions made by both. The false positives predicted by the models were quite different, which explains why the *Simple Combination* achieves the best results.

We compare the scores of the two models on each relation in the test set individually. Apart from eight relations, performance of the two models on each relation are similar. The number of false positives predicted by the *Negative Sampling* model is significantly higher than those predicted by the *Learned Tree-1 × 2* model for five relations. There are fewer positive training instances for all these five relations compared to other relations. This confirms our intuition that the *Negative Sampling* model can predict a high score for many unobserved entity pairs in relations that have few positive training instances because they are never explicitly ranked while training.

Table 3 shows the number of true positives and false positives predicted by both the models for three out of those five relations which are: /film/film/written_by, /film/film/ produced_by and /sports/sports_team_owner/teams_owned. By inspecting the false positives predicted by the *Negative Sampling* model we find that it predicts relation instances for these relations using *weak* evidence. For example, we find that it predicts the relations /film/film/written_by and /film/film/ produced_by between entity pairs that are observed with the textual surface pattern *direct→by*. Even though *direct→by* is a positive indicator for those relations it is not reliable.

The *Learned Tree-1 × 2* model predicts more false positives than the *Negative Sampling* model for three relations. Table 3 shows the number of true positives and false positives predicted by both the models for these relations. The three relations are /people/person/place_lived, /business/ person/company, /organization/parent/child. By inspecting a random subset of 30 false positives predicted by our model for each of these three relations we found that over 30% of them were actually true positives. So, we expect our model to get a higher score if evaluated on a more complete KB.

We also found the data to have many entity linking errors due to the use of simple string matching heuristic. This could affect the accuracy of both the models.

6. DISCUSSION

We develop a hierarchical model for universal schema relation extraction that maximizes the likelihood of the observed data. We avoid the need for explicit negative training data and randomly sampled "pseudo-negative" examples. We show empirically that the tree structure over the set of relations is crucial for good performance. Experimental results show that our best model attains a 2.61% gain in F1 score over the previous method described in Riedel et al. [17] and a simple combination of the two models achieves a 3.53% gain in F1 score. We expect that the gains would actually be higher if evaluated on a more complete KB.

We plan to construct a better public dataset for evaluating large-scale relation extraction approaches using Clueweb [12], which has 800 million web documents automatically annotated with 11 billion entity links to freebase. This dataset will have the following advantages over the dataset described in Riedel et al.[17]

- Its entity linking algorithm is more accurate than the string matching heuristic used in Riedel et al.[17].
- We would evaluate on a broader set of relations since the number of documents in Clueweb dataset is considerably larger than the number of NYTimes documents in the previous dataset.
- We would use a more recent version of Freebase which is more complete than the version used in Riedel et al. [17]. Although, we can never completely avoid the issue of using incomplete KBs, the documents in Clueweb were annotated with a recent version of Freebase containing several orders of magnitude more facts.

In future work, we will also consider Noise Contrastive estimation [9] to train the model which uses unobserved data as negative samples but can be shown to maximize an approximation of the likelihood of the observed data. Moreover, it does not make the assumption of a tree structure over the set of relations. We will compare it with the methods described in this paper on the new dataset.

7. ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval and in part by DARPA under agreement number FA8750-13-2-0020. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

[1] R. Bunescu and R. Mooney. Learning to extract relations from the web using minimal supervision. In *Association for Computational Linguistics*, 2007.

[2] A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In *Association for Computational Linguistics*, 2004.

[3] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *Journal of Machine Learning Research*, 2011.

[4] O. Etzioni, M. Banko, S. Soderland, and D. Weld. Open information extraction from the web. In *Association for Computing Machinery*, 2008.

[5] D. Lin and P. Pantel. Dirt -discovery of inference rules from text. In *International Conference on Knowledge Discovery and Data Mining*, 2001.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *arXiv*, 2013.

[7] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing*, 2009.

[8] A. Mnih and G. Hinton. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems*, 2009.

[9] A. Mnih and Y. W. Teh. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

[10] A. Mnih and Y. W. Teh. Learning label trees for probabilistic modelling of implicit feedback. In *Advances in Neural Information Processing Systems*, 2012.

[11] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *International Conference on Artificial Intelligence and Statistics.*, 2005.

[12] D. Orr, A. Subramanya, E. Gabrilovich, and M. Ringgaard. 11 billion clues in 800 million documents: A web research corpus annotated with freebase concepts. http://googleresearch.blogspot.com/2013/07/11-billion-clues-in-800-million.html.

[13] A. Passos, L. Vilnis, and A. McCallum. Optimization and learning in factorie. In *Advances in Neural Information Processing Systems, Workshop on Optimization for Machine Learning*, 2013.

[14] B. Recht, C. Re, S. J. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, 2011.

[15] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Conference on Uncertainty in Artificial Intelligence*, 2009.

[16] S. Riedel, L. Yao, and A. McCallum. Modeling relations and their mentions without labeled text. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, 2010.

[17] S. Riedel, L. Yao, A. McCallum, and B. M. Marlin. Relation extraction with matrix factorization and universal schemas. In *The North American Chapter of the Association for Computational Linguistics*, 2013.

[18] A. Yates and O. Etzioni. Unsupervised methods for determining object and relation synonyms on the web. In *Journal Of Artificial Intelligence Research*, 2009.