University of Texas at Austin KBP 2014 Slot Filling System: Bayesian Logic Programs for Textual Inference

Yinon Bentor **Vidhoon Viswanathan** **Raymond Mooney**
Department of Computer Science
The University of Texas at Austin
{yinon,vidhoon,mooney}@cs.utexas.edu

Abstract

This document describes the University of Texas at Austin 2014 system for the Knowledge Base Population (KBP) English Slot Filling (SF) task. The UT Austin system builds upon the output of an existing relation extractor by augmenting relations that are explicitly stated in the text with ones that are inferred from the stated relations using probabilistic rules that encode commonsense world knowledge. Such rules are learned from linked open data and are encoded in the form of Bayesian Logic Programs (BLPs), a statistical relational learning framework based on directed graphical models. In this document, we describe our methods for learning these rules, estimating their associated weights, and performing probabilistic and logical inference to infer unseen relations. Although our system was able to infer additional correct relations that were not extracted by our baseline relation extraction system, we were unable to significantly outperform a pure extraction baseline.

1 Introduction

In 2014, UT Austin was a second-time participant in the English Slot Filling task of the Text Analysis Conference (TAC) Knowledge Base Population (KBP) evaluation. In 2013, UT Austin participated in KBP Slot Filling and placed 12th out of 18 teams in official scoring, with a top F1 score of 12.26 (Bentor et al., 2013). Our 2014 system uses a similar approach, but replaces a low-performing baseline extraction system with the top performing 2013 system, RelationFactory, from the Spoken Language Group at Saarland University (with permission) (Roth et al., 2014a). Additionally, we implemented a new software architecture based on the Apache Jena Semantic Web engine that greatly improves performance and scalability of the weight learning and logical inference components of the 2013 system, and we addressed other significant shortcomings of our 2013 system.

The UT Austin KBP system aims to infer relations that are missed by a standard relation extraction system or that can be guessed using general, commonsense world knowledge. Our approach, called Bayesian Logic Programs for Textual Inference (BLP-TI) constructs Bayesian Logic Programs, a statistical relational learning formalism that combines the power of first order Horn logic with probabilistic inference using directed graphical models, to model probabilistic commonsense rules such as "the parents of a child are often spouses" or "children often live in the same state as their parents". It is our hope that such learned rules can increase the recall of relation extraction and provide useful information to other downstream systems. This work builds upon the approaches described in Raghavan et al. (2012) and Bentor et al. (2013).

2 System Architecture

The 2014 BLP-TI system architecture is shown in Figure 1. We train our system using processed relation instances from DBpedia 3.9 (Lehmann et al., 2014), as described in Section 4. An online inference-rule learning system (Raghavan and Mooney, 2013) proposes a set of first-order definite-clause rules from these relation instances, and subsequently a MLE weight learner uses counts of rela-

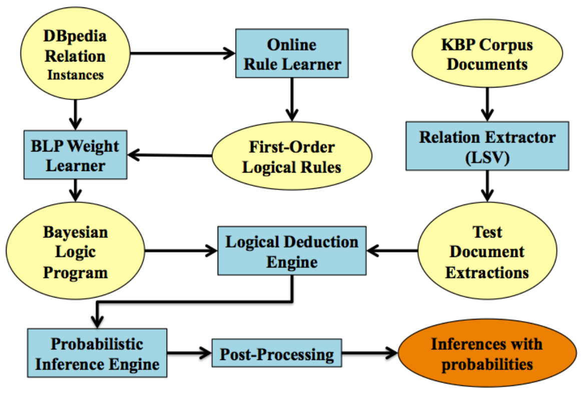

Figure 1: System overview

tion instances to assign weights to each rule. Combined, these form a Bayesian Logic Program (Kersting and De Raedt, 2007). In testing, we apply our learned BLP to extractions from an off-the-shelf relation extraction system, giving us additional inferred facts.

3 Bayesian Logic Programs

Bayesian logic programs (BLPs) are a formalism that unifies directed probabilistic graphical models and traditional Prolog-style logic programming, creating *templates* for graphical models. A BLP defines an abstract model composed of first-order Horn clauses and associated conditional probability tables (CPTs) for each rule c such that CPT(c) = P(head(c)|body(c)). In BLP clauses, all variables are universally quantified and range restricted such that $variables\{head\} \subseteq variables\{body\}$.

In BLP inference, these abstract models are instantiated with a query and a set of ground facts. Logical deduction (SLD resolution) is used to construct the graphical model by tracing the proof structure. Evidence from multiple rules is combined using a *noisy-or* combining function. Standard methods for graphical model inference in Bayes nets, such as SampleSearch (Gogate and Dechter, 2007), may then be applied.

By constructing the ground graphical model at query time and by only permitting Horn clauses in the rules, it is possible to limit the size and complexity of the ground networks, resulting in more tractable inference compared to Markov Logic Networks (MLNs) (Domingos and Lowd, 2009). Additionally, a study by Raghavan et al. (2012) shows superior performance of BLPs over MLNs on a relation inference task in a similar domain.

4 Training Data and Dependencies

Our rule learner and weight learner are both trained using relations from DBpedia. We first map relation types from the DBpedia ontology to the KBP ontology using deterministic hand-coded rules, discarding tuples that do not have a reasonable mapping in KBP. We further filter the resulting tuples to ensure that the arguments are type consistent with KBP types, resulting in approximately 1.1 million training tuples that cover 30 of the 41 KBP relation types. This increase from the 2013 UT Austin system is due to upgrading from DBPedia 3.8 to DBPedia 3.9 and an expansion of the hand-coded mapping from the DBpedia to KBP ontology.

5 Algorithms

5.1 Relation Extraction

We use an existing relation extractor for obtaining explicit relations occuring in text. Since these base relations are provided as input to our inference algorithm, it is important that they be accurate and have reasonable recall. Hence, we used the top performing *RelationFactory* system (Roth et al., 2014b) from 2013 KBP English Slot Filling Task as our base relation extractor. We will provide a brief overview of this system in this section.

The pipeline of RelationFactory system has two stages of operation. First, in the *candidate generation stage*, relevant documents are retrived and sentences that might have possible relations are filtered based on entity type checking. Second, in the *candidate validation stage*, SVM based classification is used to determine whether the candidate sentences express a valid relation for the query.

There are various tasks happening in these two stages of the pipeline. Some of the main tasks are identified and discussed briefly. A more complete and elaborate discussion can be found in Roth et al. (2014b) and Roth et al. (2012).

A major task in the first stage of pipeline is to *expand queries* as the TAC KBP query might be referenced with different alias names in training data. Hence, a reasonable set of aliases must be included for each query entity for our retrival components to return good number of candidate documents/sentences. For this, the name of query entity is expanded using Wikipedia anchor text similar to Roth and Klakow (2010).

Due to the large size of training corpus, it is important to index the documents with respect to query entities that are referenced in them. This will ensure good levels of performance of the entire relation extraction pipeline. For this purpose, we use *Apache Lucene*[1] - an efficient Java-based text search engine library. After indexing, a *point wise mutual information* based scheme is used for choosing the query aliases that need to be used for retrieving candidate documents. After retrieving relevant documents, candidate sentences are filtered by using a combination of different methods. One approach is to perform type checking based on the expected type of slot fill for each relation type. For this, *RelationFactory* uses a named entity labeler based on a perceptron-trained sequence labeler (Collins, 2002) trained on BBN training data (Weischedel and Brunstein, 2005). Another approach is to use *pattern matching* in which *RelationFactory* constructs patterns for sentences that contain specific relation slot fills and checks if a candidate sentence matches these patterns.

These filtered candidate sentences are supplied to the second stage of the pipeline, whose main task is to classify the type of relation expressed in a candidate sentence. For this purpose, they use one binary SVM classifier per relation type using SVM^{light}[2]. All candidate sentences for a particular relation form the positive training samples while the rest of the sentences form the negative samples. A set of token n-gram features are extracted from these training sentences to capture context for different relation types. *RelationFactory* uses *aggregate training* to speed up the training process by grouping all sentences by entity relation pair and using each as a single training instance. While training is done in an aggregate manner, prediction is performed on a single sentence level only. Candidate slot fills are extracted from positive predictions for each entity relation type pair, which forms one half of the input for UT Austin's BLP-TI system. In a second run, the output of the first run of *RelationFactory* is used as input, generating extractions which may be useful to satisfy the body of the *BLP-TI* rules.

[1]http://lucence.apache.org/

[2]http://svmlight.joachims.org, (Joachims, 1999)

5.2 Rule Structure Learning

Like our 2013 system, our rule learning approach follows the online approach of Raghavan and Mooney (2013), but replaces extractions from documents with sets of tuples from DBpedia.

We process 1000 relations at a time, building a directed graph whose nodes represent relation instances. Directed edges are added between relation instances that share one or more constant arguments, with edge direction chosen such that the edge's head is the more frequently seen relation instance. The edge direction encodes the heuristic that relations that are more frequently explicitly stated should help us infer relations that are less frequently stated.

After all sets of relations are processed, the rule learner traverses the resulting graph and constructs rules in which the each rule head corresponds to a tail in the graph and each rule body conjoins the nodes traversed to reach that tail. All constants in the resulting rules are replaced with unique variables to create first order rules.

Because BLPs are range restricted as described in Section 3, the rule learner retains only those rules where all variables in the head appear in the body, discarding all other non-conforming rules. The rule learner maintains a count of how often each rule is satisfied in the training set, and can be set to discard rules that do not meet a user-defined threshold.

This online rule learning algorithm has been shown to outperform a Inductive Logic Programming (ILP) system (Mccreath and Sharma, 1998) on some relation types, and, critically, to scale to much larger data sets than is possible with ILP systems.

We refer the reader to Raghavan and Mooney (2013) for additional details of the algorithm and an empirical evaluation.

5.3 Rule Weight Learning

To assign weights to the learned rules, we compute maximum likelihood estimates from the training data using a modified closed world assumption.

While we do not assume that our training set is complete, we do assume that if the training set contains facts with a particular entity occurring in the subject position, then it contains all relevant facts about that entity. (Here, by subject position we denote the first argument of the predicate.) This assumption is motivated by the observation that in linked open data resources, such as DBpedia, relation instances are often only present when a notable entity is the subject of the relation. For example, the relation $per\colon children(Barack\_Obama, Sasha\_Obama)$ is more likely to appear than $per\colon parents(Sasha\_Obama, Barack\_Obama)$, since Sasha Obama may not have her own corresponding Wikipedia page. Because of this, maximum likelihood estimates computed using the standard closed world assumption were qualitatively bad, with definitional rules (like $per\colon children(x, y) \rightarrow per\colon parents(y, x)$) assigned weights far from 1. This problem is alleviated by using the modified assumption.

Put more formally, we call a substitution *good with respect to a rule* if when that substitution is applied to the head of the rule the fact generated matches at least one fact in the training set, in both the predicate and the first argument. For each rule, we set the corresponding CPT parameter in the associated *noisy-or* combiner to be the percentage of good substitutions for that rule for which the fact generated is present in the training set.

To perform this weight learning efficiently on large data sets, we use Apache Jena, a Semantic Web Framework that can store and query large amounts of data from a knowledge base efficiently.

5.4 Inference

In testing, we use the output of two runs of the *RelationFactory* KBP system. In the first run, we obtain slot fillers for each of the KBP test queries. In a second run, we use the slot fillers found in the first run as input, generating "second-hop" results that may be able to satisfy the bodies of some of our BLP rules. For example, in the first run, *RelationFactory* may extract the value for a *per:other_family* slot for a KBP query, and in a second run, the family member would serve as a query, and *RelationFactory* may extract the value of *per:religion* for that family member. We could then use these extractions to satisfy the body of the second rule shown in Table 1.

Next, we perform BLP inference as described in (Kersting and De Raedt, 2007) and (Raghavan et al., 2012), using each KBP slot as a query on which we perform logical deduction within the Apache Jena framework, using a hybrid backward-forward reasoner. We construct a ground Bayesian network from the trace of the deduction, as described in Section 3. For each *RelationFactory* extraction, we created a "dummy node" with a combiner that incorporated the confidence reported by the extractor into the BLP. We then use SampleSearch (Gogate and Dechter, 2007), an approximate sampling method for Bayesian networks, to estimate the marginal probability of each inferred slot filler.

5.5 Post Processing

We combine extracted and inferred relation instances to form our KBP submission, preferring the relation instance with highest probability. Because this resulted in a large number of inferences, especially in the case of list valued slots, many of which had low probabilities, we construct a confidence filter and discard inferences and extractions whose confidence do not exceed a specific value. These values are learned individually for each relation type by finding the values that maximize performance on the 2013 queries.

To complete the relation provenance field (column 4) of the KBP response, we combine up to 4 relation provenance values from extractions that were used to satisfy the body of rules that were used in the deduction of an inference.

6 Results

6.1 Rules

We applied our rule learner to a set of about 1.1 million facts from DBpedia mapped to the KBP ontology using a deterministic mapping and a filter for type consistency. We divided this set into about 1000 "documents", lexicographically sorted by the first argument, and presented each segment to the rule learner in turn. We were able to map DBPedia relations to 30 of the 41 KBP predicates.

Our rule learner initially produced 627 rules, including rules with up to 3 predicates in the body.

per:country_of_birth(A,B) → per:countries_of_residence(A,B) [0.80] *If person A was born in country B, he or she likely resided in country B*
per:other_family(A,B) ∧ per:religion(B,C) → per:religion(A,C) [0.69] *If a person A is related to a person B, they may share the same religion*
per:country_of_birth(A,B) → per:country_of_death(A,B) [0.60] *If person A was born in country B, then he or she died in that country*
per:country_of_death(A,B) → per:country_of_birth(A,B) [0.66] *If person A died in country B, then he or she was likely born in that country*
per:children(A,B) → per:parents(B,A) [0.96] *If A is a child of B, B is the parent of A*
org:subsidiaries(A,B) → org:parents(B,A) [0.86] *If an organization A has a subsidiary B, B's parent organization is A*
org:parents(A,B) → org:subsidiaries(B,A) [0.58] *If an organization A's parent organization is B, B has a subsidiary A*

Table 1: Sample learned probabilistic rules. Associated learned weights are in brackets. Note that the weight on the last rule is significantly lower than its converse, which is likely because DBpedia's list of subsidiaries is often incomplete.

We filtered rules that had little empirical support (low counts in the training data) and those whose learned weights were very low. Table 1 shows some of the sample rules learned by the rule learner. We observed that many of the definitional rules had weights close to 1.0, while other rules and weights seemed qualitatively plausible. Our methods did not yield rules with more than one predicate in the body using our automated rule learner, but we supplemented the learned rules with a small set of hand-written rules. A total of 35 rules were used in our 2014 submission.

6.2 KBP Results

We submitted four runs of the UT Austin BLP-TI system, as described in Table 2. The `utaustin1` run represents our full system, combining inference and the post-process pruning to improve precision. The `utaustin4` serves as a baseline for measuring the impact of inference and pruning, while *utaustin3* isolates just the effects of pruning and `utaustin2` isolates just the effects of inference.

Because the KBP Slot Filling task does not explicitly allow for inference in relation provenance, and because providing such provenance can be tricky, the inference scores for our system have been significantly higher under lenient scoring measures such as `anydoc` and `ignoreoffsets`. Because these results were not available at the time of submission, they will be added to the proceedings version of this paper.

Run ID	**Description**
utaustin1	Full system, with inference and NIL prediction and list pruning
utaustin2	Inference system with no NIL prediction or list pruning methods
utaustin3	No inference, but apply NIL prediction and list pruning to extraction
utaustin4	The baseline run of the LSV extraction system, with no inference and no NIL prediction

Table 2: Summary of UT Austin runs

The official results are summarized in Tables 3 and 4.

7 Discussion and Future Work

From the results in Table 4, we see that inference had the effect of increasing recall by a small amount, at the expense of some precision, while the learned threshold-based pruning had the desired effect of increasing precision. Combined, however, we were unable to beat the baseline extraction performance of the *RelationFactory* system.

We were able to improve upon several deficiencies in our KBP SF 2013 submission, including the ability to infer multiple slot fillers for list-valued slots, relying on an input relation extraction system with higher performance, and exploiting the changes in KBP rules to provide additional justification off-

Run ID	Filled	Correct	KB Redundant	Response Redundant	Inexact	Incorrect
utaustin1	727	235	2	8	25	457
utaustin2	863	241	2	8	27	585
utaustin3	568	226	2	7	23	310
utaustin4	600	232	2	7	25	334

Table 3: Summary of UT Austin runs

Run ID	Diagnostic Recall	Diagnostic Precision	Diagnostic F1	Official Recall	Official Precision	Official F1
utaustin1	0.236	0.324	0.273	0.236	0.326	0.274
utaustin2	**0.242**	0.280	0.260	**0.242**	0.282	0.260
utaustin3	0.227	**0.399**	0.290	0.227	**0.401**	0.290
utaustin4	0.233	0.388	**0.291**	0.233	0.390	**0.292**

Table 4: Official results for UT Austin runs

sets that better capture the properties of inference.

The improvements in recall from performing inference were limited this year. We propose several extensions to the system as well as discuss fundamental difficulties below:

- Our system is highly dependent on a strong baseline extraction system, and the performance of existing open source systems is still somewhat low. We would like to ensemble several systems together using the BLP framework in future runs.

- Some relations are inherently difficult to infer from other relations, such as org:website or per:charges.

- Some relations can be inferred, but may be helped by an extractor that can provide extractions outside of the target ontology. For example, an extractor that targets the entire DBpedia ontology may extract relations that do not have a KBP mapping but nonetheless can be used in the body of a inference rule whose head is a KBP-mappable relation. Such an extractor would likely be trained using distant supervision methods, and will likely greatly expand the set of rules that *BLP-TI* is able to learn.

- Our rule learner is not very well suited to large datasets, as it can only consider one "document" at a time. For performance reasons, we were forced to consider sets of 1,000 relation instances from DBpedia at a time. We propose using rule learning systems such as the Path Ranking Algorithm (Lao et al., 2011), which have been shown to effectively learn rules form databases of upward of $500,000$ facts, or extensions to PRA that incorporate text corpora and distributional similarity, such as Gardner et al. (2014).

- While the KBP task definition has been expanded to allow additional justification spans, annotation guidelines currently only permit relation instances which are explicitly stated in the text or ones that can always be inferred from the text to be counted as correct. In contrast, *BLP-TI* may infer facts that are correct, but that cannot always be deduced from the justification spans given, because they are generated by probabilistic rules.

- Software problems may have limited our ability to combine inferences from several sources this year.

Similarly to our 2013 entry, our results do not quite match previous promise shown by BLPs in similar domains (Raghavan et al., 2012). BLPs show more promise than other statistical relational learning frameworks when scaling to large data sets, and we hope to show that such inferences are helpful in future tasks.

Acknowledgments

This research was supported in part by the DARPA DEFT program under AFRL grant FA8750-13-2-0026 and by MURI ARO grant W911NF-08-1-0242. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of DARPA, AFRL, ARO, DoD or the US government.

The authors would like to gratefully acknowledge the Spoken Language Systems group at the Saarland University for permission to use the *RelationFactory* KBP system. The authors would also like to acknowledge the contributions of Nick Wilson, Karl Pichotta, and Katrin Erk.

References

Yinon Bentor, Amelia Harrison, Shruti Bhosale, and Raymond Mooney. 2013. University of Texas at Austin KBP 2013 Slot Filling system: Bayesian logic programs for textual inference. In *Proceedings of the Sixth Text Analysis Conference (TAC 2013)*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

Pedro Domingos and Daniel Lowd. 2009. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan & Claypool, San Rafael, CA.

Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar., October. Association for Computational Linguistics.

Vibhav Gogate and Rina Dechter. 2007. SampleSearch: A scheme that searches for consistent samples. In *Proceedings of AISTATS 2007*.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11, pages 169–184. MIT Press, Cambridge, MA.

K. Kersting and L. De Raedt. 2007. Bayesian Logic Programming: Theory and tool. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, MA.

Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 529–539, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2014. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*.

Eric Mccreath and Arun Sharma. 1998. Lime: A system for learning relations. In *Ninth International Workshop on Algorithmic Learning Theory*, pages 336–374. Springer-Verlag.

Sindhu Raghavan and Raymond J. Mooney. 2013. Online inference-rule learning from natural-language extractions. In *Proceedings of the 3rd Statistical Relational AI (StaRAI-13) workshop at AAAI '13*.

Sindhu Raghavan, Raymond J. Mooney, and Hyeonseo Ku. 2012. Learning to "read between the lines" using Bayesian logic programs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL-2012)*, pages 349–358.

Benjamin Roth and Dietrich Klakow. 2010. Cross-language retrieval using link-based language models. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 773–774. ACM.

Benjamin Roth, Grzegorz Chrupala, Michael Wiegand, Mittul Singh, and Dietrich Klakow. 2012. Generalizing from freebase and patterns using distant supervision for slot filling. In *Proceedings of the Text Analysis Conference (TAC)*.

Benjamin Roth, Tassilo Barth, Grzegorz Chrupaa, Martin Gropp, and Dietrich Klakow. 2014a. RelationFactory: A fast, modular and effective system for knowledge base population. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 89–92, April.

Benjamin Roth, Tassilo Barth, Michael Wiegand, Mittul Singh, and Dietrich Klakow. 2014b. Effective slot filling based on shallow distant supervision methods. *arXiv preprint arXiv:1401.1158*.

Ralph Weischedel and Ada Brunstein. 2005. BBN Pronoun Coreference and Entity Type Corpus. Technical report, Linguistic Data Consortium, Philadelphia.