Probabilistic Knowledge Base assisted Question Answering

Christan Grant, Kun Li, Yan Chen, Daisy Zhe Wang
University of Florida
Computer & Information Science & Engineering Department
Gainesville, Florida
{cgrant,kli,yang,daisyw} @ cise.ufl.edu

ABSTRACT

Abstract

1. INTRODUCTION

In recent years, the large increase of machine accessible data has lead research to develop sophisticated methods of organizing and using the information. In particular, the advancement of information extraction techniques has allowed millions of facts to be extracted from across the web. This is evidenced by the renewed interests in knowledge graphs and knowledge bases from companies and researchers [1, 5, 7, 10].

Knowledge graphs efficiently store and manage the linking of facts. Knowledge bases are equivalent to knowledge graphs but with and additional inference engine. Inference engines allow the discovery of facts that are not explicitly mentioned in the knowledge graph. Researchers currently pairing probabilities with extracted facts and rules to represent the natural uncertainty found in language and extraction systems.

This demonstration describes a question answering application developed to demonstrate a new probabilistic knowledge base begin developed in the data science research group at the University of Florida. Attendees will gain an understanding of the usefulness of a probabilistic knowledge base and the techniques developed for large-scale knowledge base management. The question answering application allows users to see example queries to a probabilistic knowledge base and also to understand how probabilistic knowledge bases work in general.

Further, we enumerate our contributions as follows:

- We develop a question answering architecture that leverages a probabilistic knowledge base.
- We describe a method for users to interact with a knowledge base, adding and removing facts.
- Develop a trustworthiness value for each question answered by the knowledge base.

2. BACKGROUND

In this section, we give the necessary background from the probabilitic knowledge base backed question answering system. We first describe the set of data sets that underly this work. We then give a brief description of Markov Logic Networks followed by our definition of a probabilitic knowledgebase.

2.1 Data Set

In this demo, we use Freebase, Reverb, and NELL as the underlying knowledge bases. These knowledge bases store collection of facts as (subject, predicate, object) triples, representing facts related to well-known entities (people, places, and things). They differ in scale, schema and construction methods.

Freebase. Freebase is a web-scale, human-crafted, high precision knowledge base of well-known topics (entities) and facts [4]. As of current writing, it contains 47.5 million entities and 2.9 billion facts, spanning a variety of areas including People, Sports, Music, Internet, Books, etc, organized into domains. The Freebase KB is publicly accessible as rdf triples.

ReVerb. ReVerb is an automatic knowledge base construction system that extracts entities and facts from natural language text [8]. It is a universal schema extraction system, extracting both entities and predicates (relations). The extracted tuples are publicly available, and we use its ClueWeb extractions, which contains 14.7 million facts, annotated by their confidence and source URLs.

NELL. NELL is a never-ending knowledge extraction system [9]. It extracts facts and updates its knowledge base every day. As of current writing, it has extracted 84.6 million facts, of which 2.2 million are believed to have high confidence. Like ReVerb, each fact is annotated by its confidence and source URLs. It also contains information on which extraction algorithms are used to extract each fact.

In both ReVerb and NELL KBs, recent works have tried to mine *first-order inference rules*. An inference rule states causal relationships among facts, for example, we have

$$\mathrm{bornIn}(x, y), \mathrm{locatedIn}(y, z) \rightarrow \mathrm{bornIn}(x, z).$$

This rule states that if a person x was born in location y, and location y is located within another location z, then person x is also born in location z. The Sherlock-Holmes system mines 30,912 inference rules from ReVerb, and NELL also has 2 thousand rules with an inference engine as one of its extraction component.

2.2 Markov Logic Networks

Markov Logic Networks (MLNs) are the standard method of modeling uncertainty. MLNs are mare up of a weighted first-order formulae of the form $\{F_i, W_i\}$, where F_i is a logical expression and W_i is a weight specifying how likely it is that the formula is true.

For example, the MLN clauses below state two different sets of information.

0.96 bornInState (Obama, Hawaii)
1.40 $\forall x \in Person, \forall y \in State, \forall z \in Country :$
$bornState(x, y) \wedge isState(y, z) \rightarrow bornCountry(x, z)$

The first clause states that that Obama was born in the state of Hawaii. The second formulate is an inference rule that states that if a person x is born in a state y, and a state y is in a part of a country z, then that person x is born in the country z. These formula do not necessarily apply, the weights of 0.96 and 1.40 specify the strength of the formula; stronger rules have a lower chance of being violated. Deterministic rules, or rules that can never be violated are given an infinite weights of inf.

2.2.1 Grounding

MLNs are a template generating ground factor graphs. A factor graph is a set of factors $\Phi = \{\phi_1, \ldots, \phi_{|\Phi|}\}$, where each factor ϕ_i is a function $\phi_i(\mathbf{X}_i)$ over a vector of random variables $\mathbf{X}_i$. Maybe add figure of ground factor graph — CEG

We use the term grounding to refer to the processes of creating the factor graph from an MLN and a set of clauses. Each node in the factor graph is a ground atom and has a boolean variable that represents its truth value. We an perform the grounding step inside the database using a simple series of database queries [6].

For each possible grounding of formula F_i we create a ground factor $\phi_i(\mathbf{X}_i)$ with a value of 1 if the grounding is true, otherwise e^{W_i}. The marginal probability distribution of a set of grounded atoms $\mathbf{X}$ is defined as

$$P(\mathbf{X} = x) = \frac{1}{Z}\prod_i \phi_i(\mathbf{X}_i) = \frac{1}{Z}\exp\left(\sum_i W_i n_i(x)\right), \quad (1)$$

where $n_i(x)$ is the number of true groundings of rule F_i in x, W_i is its weight, and Z is the partition function. This probability gives use the probabilitity of one particular state of a knowledge base.

2.3 Probabilitic Knowledge Bases

We use a definition of probabilistic knowledge bases derived in previous work [6]. A probabilistic knowledge base is a 5-tuples $\Gamma = (\mathcal{E}, \mathcal{C}, \mathcal{R}, \Pi, \mathcal{L})$, where

1. $\mathcal{E} = \{e_1, \ldots, e_{|\mathcal{E}|}\}$ is the set of entities. Each entitie $e \in \mathcal{E}$ refers to a real-world object.
2. $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$ is the set of classes (or types). Each class $C \in \mathcal{C}$ maybe be a subset of $\mathcal{E} : C \subseteq \mathcal{E}$, or an unknown class.
3. $\mathcal{R} = \{R_1, \ldots, R_{|\mathcal{R}|}\}$ is the set of relations. Each $R \in \mathcal{R}$ defines a binary relation on $C_i, C_j \in \mathcal{C} : R :\subseteq C_i \times C_j$. We call C_i, C_j the domain and range of R and use $R(C_i, C_j)$ to denote the relation and its domain and range.
4. $\Pi = \{(r_1, w_1), \ldots, (r_{|\Pi|}, w_{\Pi|})\}$ is a set of weighted facts. For each $(r, w) \in \Pi$, r is a tuple (R, x, y), where $R(C_i, C_j) \in \mathcal{R}, x \in C_i \in C, y \in C_j \in C$, and $(x, y) \in R$; $w \in \mathbb{R}$ is a weight indicating how likely it is that r is true.
5. $\mathcal{L} = \{(F_1, W_1), \ldots, (F_{|\mathcal{L}|}, W_{|\mathcal{L}|})\}$ is a set of weighted rules. For each $(F, W) \in \mathcal{L}$, F is a first-order logic clause, and $W \in \mathbb{R}$ us a weight indicating how likely the formula F holds.

2.3.1 Question Answering

To answer questions, a mapping must be developed between a known natural language utterance to subset of all possible facts to present an answer. We assume that the knowledge base contains that complete set of facts necessary to find the answer to the question. Additionally, in order to quantify the truthfulness of each fact, it is important to enumerate the facts that support the final answer.

In this work, we leverage a new system named SEMPRE which uses supervised learning of question answer pairs to create a Lambda Dependency-Based Compositional Semantics language (λ-DCS) [2]. The translating to a logical form, such as a (λ-DCS), allows the semantics to be executed producing a denotation, or an answer to the question. The λ-DCS can be translated to a SPARQL query for execution over freebase of a similar knowledge base where it can be evaluated.

3. SYSTEM OVERVIEW

This demonstration describes a question answering systems that leverages a probabilistic knowledge base. To introduce this section, we first give a walkthrough of how a question is evaluated. We then give a detailed description of each component involved in question answering computation.

A user user first enters a question q. We then leverage the SEMPRE system to translate the question into SPARQL, the de facto language for querying RDF data stores. The SQL-like formalism of SPARQL queries produces a set of subgraph expressions as triples. We can then use the SPARQL query $s(q)$ and extract the supporting triples $t_{s(q)}$ that underly the subgraph. These triples are in the form $\langle s, p, o\rangle$ and correspond to the facts that support the answer to the question. To obtain the $t_{s(q)}$, we can evaluate the SPARQL query and materialize each the intermediate triple. We can use $t_{s(q)}$ to search our probabilistic knowledge base of facts to obtain the closest matching facts. Each fact $f \in \mathcal{F}$ is of the form $R(A, B)$ and a triple $t_{s(q)}$ is equivalent to a fact f when $(s, p, o) = (A, R, B)$. Given the equivalent facts, we use our k-hop algorithm to determine the joint probability that each fact is correct. This probability is a truthfulness score that can be paired with each answer to the question.

In the next few sub-sections we describe each part of the process. We start with the interface where the user can enter natural language questions and a description of how SEMPRE processes the question (Section 3.1). Next, we describe how we perform lookups for candidate facts (Section 3.2). We then define our method to derive the joint probability distribution (Section 3.3).

3.1 Interface

The interface allows users to make queries using three different modalities. Users will be able to enter natural language questions, search through the set of existing facts, and use a graph to explore connections between graphs. New probabilistic facts and rules can also be added to the system through the interface. Users can also remove or alter the existing facts and rerun queries. The status of queries

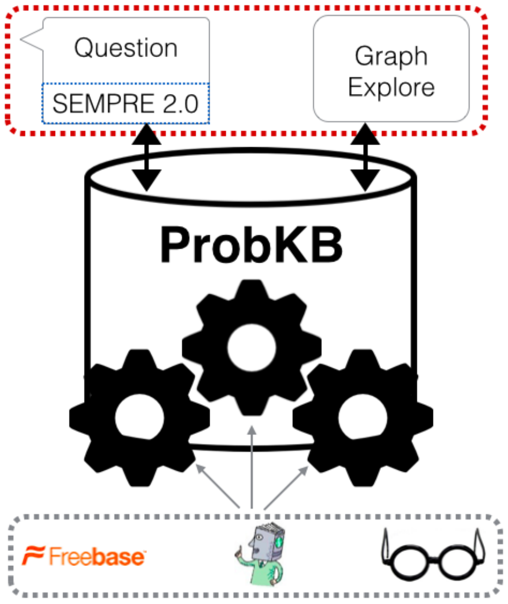


Figure 1: Question answering system architecture.

and the underlying processes are displayed on the main interface.

When a user enters a natural language utterance q, we use the SEMPRE 2.0 system to transform the utterance to a logical form [3, 2]. We then translate the logical form to SPARQL for execution over a knowledge base $\mathcal{D}$; let $s(q, \mathcal{D})$, or equivalently, $s(q)$ be a function that transforms a natural language utterance to the SPARQL query. We then parse the SPARQL query and extract the intermediate triples $t_{s(q)} = \{\langle s, p, o\rangle_1, \ldots\}$. Like the SPARQL query, these triples only specify a template of the facts that are required to evaluate the utterance. We evaluate the SPARQL query over $\mathcal{D}$ to obtain an answer α, we map the query template to define the candidate set of triples $t^{\alpha}_{s(q)}$.

For each triples in $t^{\alpha}_{s(q)}$ we perform a look up in $\mathcal{D}'$ which may or may not be equivalent to the knowledge base $\mathcal{D}'$. If there is a exact match, the triple is mapped to a score of 1. If there is no exact match in $\mathcal{D}$, we estimate the probability of the triple appearing using a straight-forward application of the chain rule. The intuition behind this weighting is that if a fact does not exist we would like to compute the probability that the facts could exist. An equation representing this value is as follows,

$$\omega(s,p,o) = \begin{cases} 1 & \text{if } \mathit{exists}(\langle s,p,o\rangle) \\ \max(\omega(o,p,s), P(s,p,o)) & \text{otherwise,} \end{cases} \tag{2}$$

where $P(s,p,o) = P(s|p,o)P(p|o)P(o)$.

We then use the K-hop algorithm to estimate the joint probability of a fact existing given the rules.

Listing 1: Algorithm for obtaining the information

```
def answer (q):
  t ={'q': q,
      's': SEMPRE.toLogicalForm (q) }
  t['a'] = SEMPRE.toSPARQL (t['s'])
  execute (D, t) # Evaluate query

  # Do fact search
  t['α'] = findMatches (D, t)
  t['ω'] = evaluateTriples (D, t) # Equation~\ref{eq:probq
  t['khop'] = kHop (D, t['α'])

  # TODO Perform a weighted combination of kHop and ω

  return t
```

```
1: procedure GetAnswer(q)
2:     $t \leftarrow \{q : q\}$
3:     $t \leftarrow \{s : \text{toLogicalForm}(q)\}$
4:     $t \leftarrow \{a : \text{toSPARQL}(t.s)\}$
5:     $t \leftarrow \{res : \text{execute}(\mathcal{D}, t.a)\}$
```
Algorithm 1: Question answering algorithm.

3.2 Fact Search

Given a candidate fact f of the form $\langle s, p, o\rangle$, we use the database to search for the top triples that are similar to f. Some sets of facts contain blank nodes or expect lists or sets of information. For example, the utterance "`Who are Justin beiber's siblings?`" produces a SPARQL query that contains the following subtriples: (1)

Describe D3 visualization of graph and rule display Describe user interaction with graph Describe user selecting facts Describe users removing facts

3.3 Probabilistic Inference

Given that we have a set of matching/partially matching facts, give an explanation of how values from the khop algorithm are evaluated.

We compare the fact probabilities to the sempre results.

3.3.1 Knowledge Base

Describe the PostgreSQL database and the other services running on servers. Describe the tables Describe the functions that are called Describe the parallelism

The system is loaded with docker, a system container, so any modifications by demo can be quickly rolled back to the initial state.

4. RELATED WORK

Describe incremental KB from Chris Re

Question answering over freebase [11].

Describe Google Knowledge Vault Describe Fact finding in Google KB Describe Source finding.

5. DEMONSTRATION

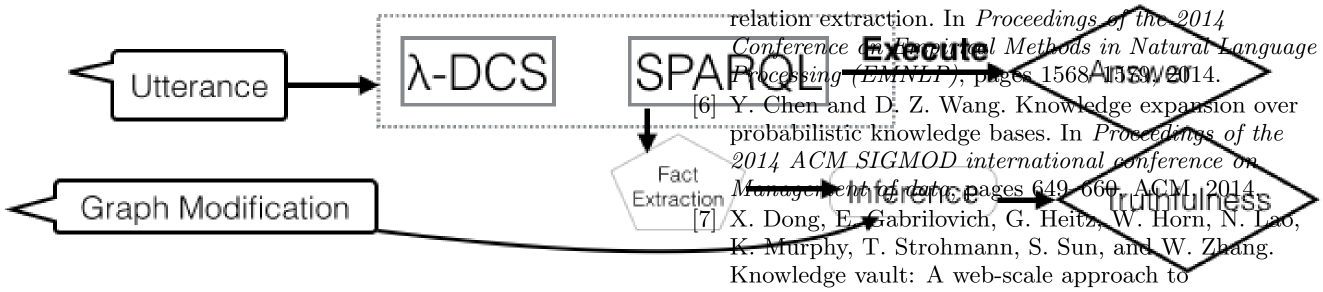
Describe the demo setup.

Features Add the auto complete for previous questions.

Describe how users will be able to alter parameters. Add figure of the pipeline process and the user interface.

6. ACKNOWLEDGMENTS

This work was partially supported by DARPA under FA8750-12-2-0348-2 (DEFT/CUBISM) and Christan Grant was supported by a NSF Graduate Research Fellowship, Grant DGE-0802270.



H

Figure 2: Probabilistic knowledge base assisted question answering demonstration pipeline.

7. REFERENCES

[1] K. Bellare, C. Curino, A. Machanavajihala, P. Mika, M. Rahurkar, and A. Sane. Woo: A scalable and multi-tenant platform for continuous knowledge base synthesis. *Proceedings of the VLDB Endowment*, 6(11):1114–1125, 2013.

[2] J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, pages 1533–1544, 2013.

[3] J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. 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, 2008.

[5] K.-W. Chang, W.-t. Yih, B. Yang, and C. Meek. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1568–1579, 2014.

[6] Y. Chen and D. Z. Wang. Knowledge expansion over probabilistic knowledge bases. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 649–660. ACM, 2014.

[7] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM, 2014.

[8] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference of Empirical Methods in Natural Language Processing (EMNLP '11)*, Edinburgh, Scotland, UK, July 27-31 2011.

[9] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, 2015.

[10] F. Niu, C. Zhang, C. Ré, and J. W. Shavlik. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. *VLDS*, 12:25–28, 2012.

[11] X. Yao and B. Van Durme. Information extraction over structured data: Question answering with freebase. In *Proceedings of ACL*, 2014.