A Demonstration of Sya: A Spatial Probabilistic Knowledge Base Construction System

Ibrahim Sabek[1], Mashaal Musleh[1], Mohamed F. Mokbel[1,2]
[1]Department of Computer Science and Engineering, University of Minnesota, USA
[2]Qatar Computing Research Institute, HBKU, Qatar
{sabek,musle005,mokbel}@cs.umn.edu

ABSTRACT

This demo presents *Sya*; the first full-fledged spatial probabilistic knowledge base construction system. *Sya* is a comprehensive extension to the DeepDive [18] system that enables exploiting the *spatial relationships* between extracted relations during the knowledge base construction process, and hence results in a better knowledge base output. *Sya* runs existing DeepDive programs as is, yet, it extracts more accurate relations than DeepDive when dealing with input data that have spatial attributes. *Sya* employs a simple spatial high-level language, a rule-based spatial SQL query engine, a spatially-indexed probabilistic graphical model, and an adapted spatial statistical inference technique to infer the factual scores of relations. We demonstrate a system prototype of *Sya*, showing a case study of constructing a crime knowledge base. The demonstration shows to the audience the internal steps of building the knowledge base, as well as a comparison with the output of DeepDive.

KEYWORDS

Knowledge Base Construction, Spatial Knowledge Bases

1 INTRODUCTION

Knowledge base construction has been an active area of research over the last two decades with several system prototypes coming from academia (e.g., [9, 19]) and industry (e.g., [10, 14, 17]), along with many important applications, e.g., web search [2], digital libraries [1], and health care [11]. The goal of knowledge base construction is to extract *factual* structured data from unstructured data sources, e.g., Wikipedia, semantic web, and business logs. Example of such facts include "Alice married Bob" or "John has Ebola". Most recently, the idea of *probabilistic* (instead of *factual*) knowledge bases has been proposed, where each extracted *relation* is associated with a probability of how the system is confident that this relation is factual (e.g., see [12, 13, 16, 18]). Examples of such probabilistic relations is "Alice married Bob with probability 80%". Among existing probabilistic knowledge base systems, DeepDive [18] has achieved more popularity and widespread adoption in vital applications like fighting human trafficking [3], and building several domain-specific knowledge bases (e.g., [20, 21]).

DeepDive has no support for exploiting the location information associated with extracted relations, however, such information increases the accuracy of output knowledge bases. For example, DeepDive could assign a high factual score for the relation "Alice married Bob" due to its extensive mention in input data. However, if the residence locations of "Alice" and "Bob" are very far (e.g., different countries), then such factual score should be lowered regardless of how much support for this relation in input data. Another common example in spatial analysis applications such as epidemic diseases studies, a relation "John has Ebola" with high factual score implies that another relation "Bob has Ebola" should have high score as well due to the spatial closeness between "John" and "Bob" (e.g., same neighborhood), even if the input records of "Bob" have no Ebola symptoms. Such spatial relationships within/between extracted relations motivate an interesting new paradigm of spatial-aware knowledge bases, whereby DeepDive should exploit the spatial aspect of relations during its construction process.

This demo presents *Sya*; a full-fledged spatial probabilistic knowledge base construction system, built in the core of DeepDive. *Sya* works in a similar way to DeepDive, and hence existing DeepDive programs, can run as is on *Sya*. Yet, if the programs deal with input data that have spatial attributes, *Sya* will extract more accurate relations. In general, *Sya* supports the spatial awareness inside the three main modules of DeepDive, namely, *language*, *extraction*, and *inference* modules. In the *language* module, *Sya* extends the high-level language in DeepDive to support spatial data types, predicates and parameters. To accommodate for the newly introduced spatial constructs in the language, *Sya* extends the *extraction* module in DeepDive to evaluate rules with spatial constructs as spatial SQL queries. In the *inference* module, *Sya* introduces a spatially-indexed probabilistic graphical model and a spatial inference technique, that both exploit the spatial relationships between extracted relations when inferring their factual scores.

We demonstrate *Sya* with its real system prototype, showing a use case of building a knowledge base about crime rates in Minneapolis, MN. In addition to inspecting the output relations with their factual scores and comparing them to a ground truth, the demonstration will allow the audience to examine the intermediate steps of constructing the knowledge base, experience the effect of adding/removing rules, and observe the accuracy difference between *Sya* and DeepDive.

2 SYA ARCHITECTURE

Figure 1 depicts *Sya* system architecture, which is similar to DeepDive, yet, with spatial extensions. There are two types of users who interact with *Sya*, *domain experts* and *casual users*. Domain experts are advanced users that use the provided high-level language by *Sya* to express the knowledge base construction logic. Casual users are non-technical users who access the final knowledge base output through either querying or visualization APIs. *Sya* input can be structured (e.g., DB relations) and/or unstructured (e.g., text) data.

Sya extends the three main modules in DeepDive, namely, *language*, *extraction*, and *inference* modules. In the *language* module,

[1]This work is partially supported by the National Science Foundation, USA, under Grants IIS-1525953, and CNS-1512877.

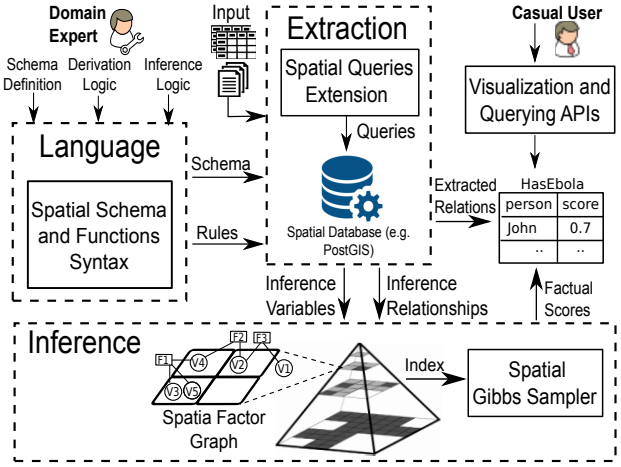

Figure 1: *Sya* System Architecture.

Sya extends DDlog [18], the declarative language of DeepDive, to support spatial data types, functions and parameters in its schema and rules syntax. In the *extraction* module, *Sya* extends the query engine in DeepDive to support translating and executing the incoming schema and rules from the language module, when they have spatial constructs, as spatial SQL queries. There are two types of queries that are generated from this module. The first type extracts relations themselves. The second type initializes variables that will be used in the inference module to infer factual scores of the relations (referred to as *inference variables*), and defines how values of these variables depend on each other (referred to as *inference relationships*). In the *inference* module, *Sya* infers the factual scores of relations using (1) a new data structure, referred to as *Spatial Factor Graph*, that spatially organizes inference variables and their relationships and (2) an adapted inference technique that exploits this new data structure during its processing.

3 THE LANGUAGE MODULE

Figure 2 shows an example of a *Sya* program. Similar to DDlog [18], the program consists of three types of syntax, namely, *schema declaration*, *derivation rules* and *inference rules*. *Sya* adds spatial constructs to each type of syntax.

Schema Declaration. *Sya* extends the schema of relations in DDlog with three spatial data types, namely, `Point`, `Rectangle`, and `Polygon`. For example, the schema of the *Person* relation consists of two non-spatial attributes; `id` and `hasVirus`, and one spatial attribute; `location` of `Point` data type. *Sya* allows the schema corresponding to the extracted relations, ended with question mark, to contain one spatial attribute at maximum. Such attribute will be used during the inference of factual scores later (details are described in Section 5). For example, the schema declaration of the *HasEbola* relation, which holds extracted relations about people infected with Ebola (e.g., "John has Ebola"), has only one spatial attribute `location` that holds the locations of persons satisfying these relations (e.g., "John").

Derivation Rules. *Sya* extends the predicates of *derivation* rules in DDlog to include three spatial primitive functions, namely,

```
#Schema Declaration
Person (id bigint, location Point, hasVirus bool).
HasEbola? (id bigint, location Point).

#Derivation Rule
HasEbola(P1, L1) = NULL :- Person(P1, L1, -), Person(P2, L2, F2),
                [Distance(L1, L2) < 10, F2 = true].

#Inference Rule
@weight (10), @locality (5)
HasEbola(P1, -) => HasEbola(P2, -) :-
                Person(P1, -, -), Person(P2, -, -).
```

Figure 2: Example on *Sya* Language.

`Overlaps` to test whether two shapes overlap or not, `Within` to test whether a shape is in a certain range or not, and `Distance` to calculate the distance between the centers of two shapes. Figure 2 shows an example of a rule to derive the *HasEbola* relation based on a spatial predicate. In this example, a relation is extracted for each person who has another person with Ebola virus and is located within 10 miles neighborhood. Note that this rule is implemented as a self join on the *Person* relation with equality (`F2=true`) and distance (`Distance(L1,L2)< 10`) predicates.

Inference Rules. *Sya* overrides the first-order logic symbols [15] in DDlog inference rules (e.g. imply, negate), when their arguments have relations with spatial attributes, to exploit the *spatial relationships* between extracted relations when inferring their factual scores (details are described in Section 5). In addition, *Sya* extends the parameters of inference rules with the `@locality(N)` parameter to specify the level of spatial neighbourhood applied on relations, where `N` is the level value. Figure 2 shows an example of an "imply" inference rule over the extracted relations in *HasEbola*. The rule states that factual scores of each pair of extracted Ebola relations affect each other. Such effect increases if these relations are located within the same spatial neighbourhood.

4 THE EXTRACTION MODULE

To accommodate for the newly introduced spatial constructs, e.g., `Distance`, in rules, *Sya* extends the query engine in DeepDive to translate rules with such constructs as a set of *range query* and *spatial join* queries. Then, these queries are executed through standard spatial database engines, e.g. PostGIS [6], MySQL Spatial [7]. Such engines support both spatial and non-spatial queries. Thus, original SQL queries in DeepDive can still be executed on them. To enable any spatial database engine to run inside *Sya*, the database driver interface in DeepDive is extended to support defining spatial storage, function and query capabilities. Currently, *Sya* provides a full integration with PostGIS as a case study.

5 THE INFERENCE MODULE

Sya inference module is the main component that is responsible on estimating the factual scores of extracted relations. Unlike DeepDive, *Sya* suggests that *spatially* close extracted relations have higher correlation (i.e., highly depend on each other) in their factual scores more than distant ones. Assume three extracted relations "John has Ebola", "Bob has Ebola" and "Alice has Ebola", where

"John" is spatially closer to "Bob" than "Alice", then the factual score of "John has Ebola" will correlate with "Bob has Ebola" more than "Alice has Ebola". We refer to this concept as *inference locality*. To efficiently apply this concept when inferring factual scores of extracted relations, *Sya* introduces a spatially-indexed probabilistic graphical model (Section 5.1), called *Spatial Factor Graph*, as well as an efficient variation of an inference algorithm (Section 5.2), called *Spatial Gibbs Sampler*, that employs this graphical model.

5.1 Spatial Factor Graph

Sya builds a spatial index for factor graph [22], which is a graphical model used to represent inference variables (i.e. random variables that are used to estimate factual scores) and their relationships in the inference algorithm. The index details are described below.

Index Structure. *Sya* employs an in-memory partial pyramid index [8]. The pyramid index partitions the whole space into grid cells with different granularity L (pyramid levels). In each level, the same factor graph is partitioned. This means that we have L spatially-indexed factor graphs, however, in different granularities. Within each cell, *Sya* indexes inference variables (each variable corresponds to one relation) contained in this cell, based on their spatial attribute values, i.e. associated spatial objects. A variable is replicated in many cells if its spatial object overlaps with them. Each variable is associated with a hash index of pointers to other connected variables (whether they are inside the same cell or not) from the factor graph.

Index Maintenance. Since the knowledge base construction is an incremental process, more relations are extracted from time to time, and hence their corresponding variables are added to the index in *bulk*. *Sya* adapts the index structure based on the incoming variables via split and merge operations. As splitting and merging pyramid cells are expensive operations, they are done only if necessary. Basically, four cells are merged only if three of them are empty. A cell is split only if it is over capacity and splitting its contents span at least two children cells. If variables are no longer needed, they are deleted in a *lazy* way.

5.2 Spatial Gibbs Sampler

Sya adapts a variation of Gibbs sampling [22] algorithm, an approximate inference algorithm used in DeepDive, to apply the inference locality concept. Given the spatial factor graph G, the algorithm performs the following three main steps:

- For each variable V (corresponding to an extracted relation), the locality parameter (`locality(N)`) specified in the language is first used to find the cell in the pyramid index of G that contains V. If this cell is not maintained, the nearest maintained ancestor cell is retrieved. Then, all neighboring variables that are connected to V in that cell are retrieved.
- A Gibbs sampling [22] iteration is applied such that the correlation between the values of V and its neighbors, obtained from the first step, are *maximized*.
- The first and second steps are repeated many times. After enough iterations over all variables, their scores are calculated using the law of total probability [22].

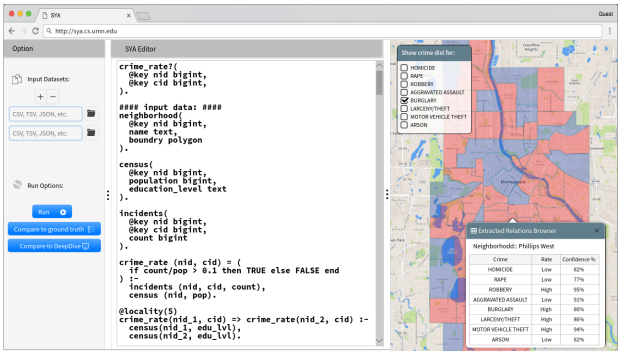

Figure 3: Relations Extraction using *Sya*.

6 DEMO SCENARIO

In this section, we demonstrate an example application of building a knowledge base about crime rate in Minneapolis, MN using *Sya*. In addition, we demonstrate comparison scenarios that show the accuracy of *Sya* over DeepDive in building the same knowledge base. In particular, we use both *Sya* and DeepDive to build a knowledge base about the rate of 8 crime types (e.g., robbery, rape and homicide), in 87 neighborhoods in Minneapolis. We use three types of structured datasets: (1) *neighborhood* dataset that contains 87 entries about neighborhood names and their geographical boundaries. (2) *census* dataset that contains 87 entries, corresponding to all neighborhoods in Minneapolis, about population count and their education levels. (3) *incidents* dataset that contains 22382 entries about individual crime incidents and their locations in 50 neighborhoods only. The first and third datasets are collected from the official Minneapolis government data hub (OpenDataMinneapolis) [5], while the second dataset is from [4]. Our demo attendees would be able to interact with *Sya* through one or more of the following scenarios.

6.1 Scenario 1: Interactive Relations Extraction

In this scenario, we show the powerful ability of *Sya* to *interactively* extract knowledge base relations and their factual scores, where the results are changing instantly based on the rules that demo attendees would feed to the system.

Figure 3 shows the main user interface of *Sya*. The user interface facilitates the user to provide: (1) input datasets, which are in Text, CSV, TSV or any other format originally supported by DeepDive, through the left most panel, and (2) input schema declaration and derivation/inference rules through the middle text editor. The figure shows an example of a sample program written in *Sya* language for extracting knowledge base about the rates (whether *high* or *low*) of 8 crime types, in each neighborhood. In particular, the program has a main inference rule which relates the crime rates of different neighborhoods based on the education level of their population, while taking into account the spatial relationships (i.e., inference locality) between the neighborhoods.

To execute the program, the user should click on the "Run" button in the most left panel. The application then submits this program to *Sya* back-end to produce the output relations and their factual scores. The interface displays the output relations as thematic map, where the user can display the results for a certain crime type (e.g.,

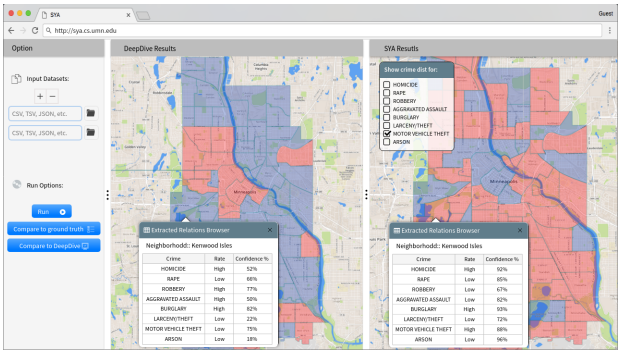

Figure 4: Visualizing comparison of *Sya* and DeepDive.

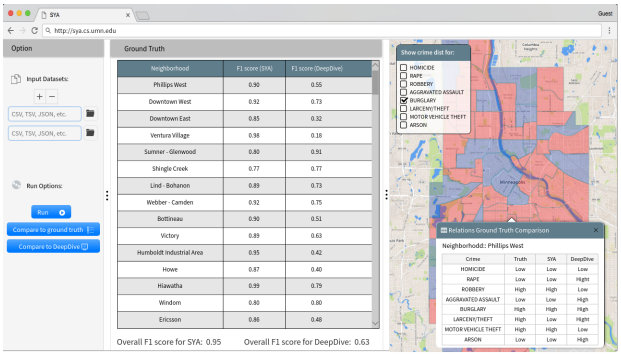

Figure 5: Ground truth comparison of *Sya* and DeepDive.

"BURGLARY") in all neighborhoods by selecting its name from a check box list on the right part of the interface. For each neighborhood, the rate of this crime type is then displayed in either red (corresponds to high) or blue (corresponds to low) colors as shown in the figure. To obtain a summary about crime types in a specific neighborhood, the user should double click its corresponding polygon on the map. As a result, a pop-up window will appear to show the rate and confidence of each crime type in this neighborhood.

6.2 Scenario 2: Comparison with DeepDive

To show the accuracy of *Sya*, we provide a scenario for comparing the crime rate knowledge base output of both *Sya* and DeepDive. In case of DeepDive, we write a program similar to *Sya*, however, without any spatial support. Demo attendees can see the two programs for comparison. The user interface provides two ways for visualizing the comparison between two systems. The first way is to show their outputs on two different thematic maps through clicking the "Compare to DeepDive" button, where each map displays the same types of information as described in Scenario 1. Figure 4 shows an example of the two generated maps in case of selecting the crime type as "MOTOR VEHICLE THEFT". The second way is to calculate their F1-measure scores based on the ground truth by clicking the button "Compare to ground truth". Figure 5 shows the F1-measure scores of both systems displayed for each neighborhood. It shows F1-scores over all neighborhoods as well.

6.3 Scenario 3: System Internals

Monitoring the system internals would be of interests for demo attendees as well as system developers. It can help in understanding the components of the system, identifying the performance bottlenecks and even allowing users to improve its design. Thus, in our demo, we show the attendees the intermediate outputs of the extraction and inference components (e.g., the generated spatial SQL queries). In addition, we show attendees the steps of installing *Sya*, on both single and multiple machines scenarios, and configuring its dependencies, e.g., configuring PostGIS to be used in *Sya*.

REFERENCES

[1] CiteSeerX. http://citeseerx.ist.psu.edu/.
[2] Google Knowledge Graph. https://www.google.com/intl/en-419/insidesearch/features/search/knowledge.html.
[3] Human Trafficking. https://www.cbsnews.com/news/new-search-engine-exposes-the-dark-web/.
[4] MinnesotaCompass. http://www.mncompass.org/.
[5] OpenDataMinneapolis. http://opendata.minneapolismn.gov/.
[6] PostGIS. http://postgis.net/.
[7] SpatialLite. https://www.gaia-gis.it/fossil/libspatialite/index.
[8] Walid G. Aref and Hanan Samet. Efficient Processing of Window Queries in the Pyramid Data Structure. In *PODS*, 1990.
[9] Soren Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A Nucleus for a Web of Open Data. In *ISWC-ASWC*, 2007.
[10] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *SIGMOD*, 2008.
[11] Ranjit Bose. Knowledge Management-enabled Health Care Management Systems. *Expert Systems with Applications*, 2003.
[12] Yang Chen and Daisy Zhe Wang. Knowledge Expansion over Probabilistic Knowledge. *SIGMOD*, 2014.
[13] Yang Chen, Xiaofeng Zhou, Kun Li, and Daisy Zhe Wang. Archimedes: Efficient Query Processing over Probabilistic Knowledge Bases. *SIGMOD Record*, 2017.
[14] Omkar Deshpande, Digvijay S. Lamba, Michel Tourn, Sanjib Das, Sri Subramaniam, Anand Rajaraman, Venky Harinarayan, and AnHai Doan. Building, Maintaining, and Using Knowledge Bases: A Report from the Trenches. In *SIGMOD*, 2013.
[15] Michael Genesereth and Nils Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 1987.
[16] Gjergji Kasneci, Maya Ramanath, Fabian Suchanek, and Gerhard Weikum. The yago-naga approach to knowledge discovery. *SIGMOD Record*, 2009.
[17] Rajasekar Krishnamurthy, Yunyao Li, Sriram Raghavan, Frederick Reiss, Shivakumar Vaithyanathan, and Huaiyu Zhu. SystemT: A System for Declarative Information Extraction. *SIGMOD Record*, 2009.
[18] Jaeho Shin, Sen Wu, Feiran Wang, Christopher De Sa, Ce Zhang, and Christopher Ré. Incremental Knowledge Base Construction Using DeepDive. *PVLDB*, 2015.
[19] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In *WWW*, 2007.
[20] M Whirl-Carrillo, E M McDonagh, J M Hebert, L Gong, K Sangkuhl, C F Thorn, R B Altman, and T E Klein. Pharmacogenomics Knowledge for Personalized Medicine. *Clinical Pharmacology and Therapeutics*, 2012.
[21] Ce Zhang, Vidhya Govindaraju, Jackson Borchardt, Tim Foltz, Christopher Ré, and Shanan Peters. GeoDeepDive: Statistical Inference Using Familiar Data-processing Languages. In *SIGMOD*, 2013.
[22] Ce Zhang and Christopher Ré. Towards High-throughput Gibbs Sampling at Scale: A Study Across Storage Managers. In *SIGMOD*, 2013.