Adopting Markov Logic Networks for Big Spatial Data and Applications

Ibrahim Sabek
Supervised by: Mohamed F. Mokbel
Department of Computer Science and Engineering
University of Minnesota, MN, USA
{sabek, mokbel}@cs.umn.edu

ABSTRACT

Spatial data has become ubiquitous everywhere, e.g., GPS data, medical data, with increasingly sheer sizes. This raises the need for efficient spatial machine learning and analysis solutions to extract useful insights from such data. Meanwhile, Markov Logic Networks (MLN) have emerged as powerful framework for building usable and scalable machine learning tools. Unfortunately, MLN is ill-equipped for spatial applications because it ignores the distinguished spatial data characteristics. This paper describes SMLN, the first full-fledged MLN framework with native support for spatial data. SMLN comes with a high-level datalog-like language with spatial constructs, and spatially-equipped grounding, inference and learning modules. We show the effectiveness of SMLN by illustrating three systems, namely, Sya, TurboReg, and Flash, that are already built using SMLN.

1. INTRODUCTION

Data scientists and developers have been spending significant efforts applying machine learning and artificial intelligent methods, e.g., deep learning, to analyze and turn their massive data into useful insights. However, the expertise skills and efforts needed to deploy these methods become a major blocking factor in having a wide deployment of machine learning applications. As a result, Markov Logic Network (MLN) [19] was recently introduced to reduce this gap. In particular, MLN combines first-order logic [5] with probabilistic graphical models to efficiently represent statistical learning and inference problems with few logical rules (e.g., rules with imply and bit-wise AND predicates) instead of thousands of lines of code. With MLN, data scientists and developers do not need to worry about the underlying machine learning work. Instead, they will focus their efforts on developing the rules that represent their applications. Recently, MLN has been widely adopted as a research vehicle for deploying machine learning in various applications, including knowledge base construction [25], data cleaning [18], genetic analysis [23], among others.

Meanwhile, in recent years, there has been a proliferation in the amounts of spatial data produced from several devices such as satellites, space telescopes, and medical devices. Various applications and agencies need to analyze these unprecedented amounts of spatial data. For example, epidemiologists use spatial analysis techniques to track infectious disease [2]. News reporters use geotagged tweets for event detection and analysis [24]. Unfortunately, researchers never take advantage of the recent advances of Markov Logic Networks (MLN) to boost the usability, scalability, and accuracy of spatial machine learning tasks (e.g., spatial regression [9]) used in these applications. Furthermore, MLN-based applications (e.g., knowledge base construction [25]) would miss important results and have less accuracy when dealing with spatial data. The main reason is that MLN is oblivious to the spatial data. The only way to support spatial data in MLN is to simply ignore its distinguished properties (e.g., spatial relationships among objects) and deal with it as non-spatial data. While this would work to some extent, it will result in a sub-par performance.

The goal of our work is to provide the first full-fledged MLN framework with a native support for spatial data, called *Spatial Markov Logic Networks* (SMLN). In particular, SMLN pushes the spatial awareness inside the internal data structures and core learning and inference functionalities of MLN, and hence inside all MLN-based machine learning techniques and applications. SMLN consists of four main modules, namely, *language*, *grounding*, *inference* and *learning*. The *language* module extends the DDlog language [25] with spatial data types and predicates to express spatial semantics when writing rules. The *grounding* module constructs a spatial variation of the factor graph [28], namely *Spatial Factor Graph*, to efficiently represent SMLN graphical models. The *inference* module provides a novel algorithm of Gibbs Sampling [30] that combines the Conclique [11] concept from spatial statistics with query-driven independent Metropolis-Hastings approach [12]. The *learning* module employs a new distance-based optimization technique based on the gradient descent method [31] to efficiently learn SMLN model parameters.

Users of SMLN would be able to seamlessly build a myriad of scalable spatial applications, without worrying about the underlying spatial machine learning and computation details. We show three case studies that use SMLN as a backbone for their computation. These case studies include

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

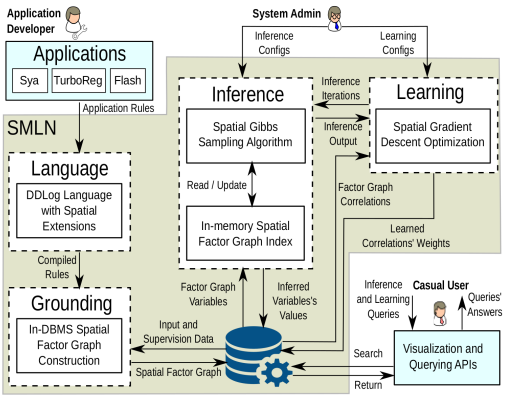


Figure 1: SMLN System Architecture.

Sya [21], a system for spatial probabilistic knowledge base construction, TurboReg [20], a framework for scaling up spatial autologistic regression models, and Flash [22], a framework for scalable spatial data analysis.

2. SMLN OVERVIEW

Figure 1 provides an overview of SMLN architecture. There are three types of users who interact with SMLN, namely, *developers*, *casual users*, and *administrators*. *Developers* should have expertise with MLN and use the provided high-level language by SMLN to create new applications. We briefly review three example applications including Sya [21], TurboReg [20], and Flash [22] in sections 3, 4 and 5, respectively. *Casual users* can either use standard querying or visualization APIs to perform inference and learning queries over the built applications (e.g., what is the probability of a specific event to happen?). *Administrators* can monitor the system and tune up the inference and learning configurations. SMLN adopts an extensible approach, where it injects the spatial awareness inside the four main modules of MLN, namely, *language*, *grounding*, *inference*, and *learning*. In the rest of this section, we highlight our contributions in each of these four modules.

2.1 The Language Module

SMLN employs a high-level language to help users write on-top applications as a set of rules and save huge coding efforts. Instead of providing a completely new language, SMLN extends DDlog [25], a datalog-like language for defining MLN rules, with spatial data types and predicates that conform to the Open Geospatial Consortium (OGC) standard [16]. Such extensions allow users to express their spatial semantics without the need for re-implementing user-defined functions in each application. For example, SMLN adds four spatial data types, namely, `point`, `rectangle`, `polygon`, and `linestring`, to the schema declaration of relations in DDlog. Figure 2 shows an example of two schema declarations $S1$ and $S2$ with `point` spatial attributes. In addition, SMLN extends the *derivation*, *supervision* and *inference* rules in DDlog with spatial predicates (e.g., `overlaps`, `within`, and `distance`) and functions (e.g., `union` and `buffer`) to efficiently evaluate the relationships between spatial objects. Such predicates and functions can be composed. For example, the inference rule $R1$ in

Schema Declaration
S1: County (id bigint, location **point**, hasLowSanitation bool).
S2: HasEbola? (id bigint, location **point**).

Inference Rule
R1: HasEbola(C1, L1) => HasEbola(C2, L2) :-
County(C1, L1, -), County(C2, L2, S2)
[**distance**(L1, L2) < 2.5, **within**(liberia_geom, L1), S2 = true].

Figure 2: Example on Spatial Extensions in DDlog.

Figure 2, which captures the effect of Ebola infected counties on each other, is composed of two spatial predicates `distance` and `within` that measure the distance between infected counties, and check whether they are located in Liberia or not, respectively. Once submitting the SMLN program, this module checks the syntax correctness of used spatial constructs, compiles the program, and forwards the output to the *grounding* module.

2.2 The Grounding Module

Grounding is an essential operation in the MLN execution pipeline, where it constructs a data structure called factor graph [28] that will be used later to perform inference and learning operations on. Such factor graph is efficiently constructed by evaluating the compiled rules from the *language* module as a sequence of SQL queries (e.g., [25]). To accommodate for the newly introduced spatial constructs, e.g., `distance`, in rules, SMLN adapts a scalable in-database grounding technique from [14] to translate and evaluate rules with these constructs as a set of spatial SQL queries (e.g., range query and spatial join). The generated queries are then executed through standard spatial database engines, e.g. PostGIS, to produce a spatial variation of the factor graph, namely, *Spatial Factor Graph*, that consists of: (1) logical and spatial random variables. (2) logical and spatial correlations among these variables.

SMLN provides two effective optimizations in the grounding process: (1) It supports creating on-fly spatial indices (e.g., R-tree [7]) on relations with spatial attributes, making the evaluation of complex predicates (e.g., `overlap`) more efficient. (2) It provides a simple heuristic query optimizer that re-orders the execution of nested spatial queries that come from rules with multiple spatial predicates. Moreover, SMLN provides an abstract database driver that supports defining spatial storage, functions and query capabilities. Such abstract can be extended by users to run their spatial database engine choice inside SMLN.

2.3 The Inference Module

The main objective of the inference module in MLN is to infer the values of variables in the constructed factor graph and compute their associated probabilities in an efficient and scalable manner [14]. Gibbs Sampling [30] is considered the most widely used inference algorithm in MLN systems, mainly because its simplicity and efficiency. However, there are two main limitations in using the existing Gibbs sampling techniques when inferring the values of the spatial factor graph variables. First, these techniques infer values that maximize the satisfaction of the logical semantics (e.g., imply) encoded in the factor graph. Therefore, in case of spatial factor graph, these inferred values will be suboptimal because they never consider the spatial correlation between variables. Second, these techniques require a large num-

ber of sampling iterations (i.e., slow convergence) to obtain an acceptable output in case there are spatial correlations among variables, because they perform sequential sampling over the factor graph nodes [11].

To overcome the above two limitations, SMLN employs a novel Gibbs Sampling algorithm, namely *Spatial Gibbs Sampling*, that can efficiently perform inference on the spatial factor graph coming from the *grounding* module. To take the spatial correlations into account, the proposed sampling algorithm adapts a new variation of the query-driven independent Metropolis-Hastings approach [12] that uses inverse-distance method [13] to spatially weigh the correlations among variables in the spatial factor graph, and hence yields more accurate inferred values.

To alleviate the slow convergence issue, a straightforward solution is to randomly partition the variables into a set of buckets and then sample these buckets in parallel. Even though this solution will finish the sampling iterations faster than the sequential one, it may not converge to an acceptable solution as spatially-dependent variables might run in parallel (i.e., independent of each other). This will force the sampler to run additional sampling iterations to converge, and hence incur a significant latency overhead. As a result, SMLN employs an approach that combines in-memory spatial partitioning technique, namely pyramid index [1], with a well-known spatial statistics concept, namely concliques [11], to heuristically partition the spatial factor graph into a set of spatially-independent partitions, and sample them in parallel. Defining concliques ensures the neighbouring independence between nodes in the same conclique set, and hence these nodes can be efficiently sampled in parallel.

2.4 The Learning Module

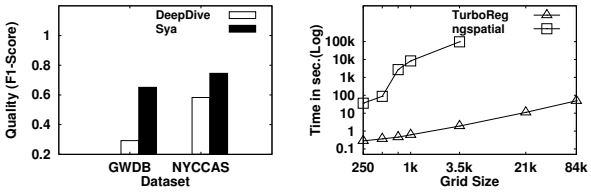
In general, the learning module in MLN mainly focuses on optimizing the weights of correlations defined in the factor graph. However, in case of spatial factor graph, the relative distance between spatial variables participating in any correlation should be considered as well to learn optimal weights. In particular, correlations between spatially close variables should have higher effect on learned weights than correlations between distant variables. We refer to this concept as *correlation locality*. Recently, the gradient descent optimization [31] has been widely used in optimizing the weights in MLN models that use Gibbs sampling inference (e.g., [25]). However, standard gradient descent optimization techniques fall short in supporting the correlation locality concept. As a result, SMLN introduces a new variation of gradient descent optimization that applies distance-based weighing technique. Given a correlation c, defined over m spatial variables $v_1, v_2, ..., v_m$ in the spatial factor graph, its weight w_c is optimized as follows:

$$w_c = w_c + \frac{m(m-1)}{2\sum_{i=1}^{m-1}\sum_{j=i+1}^{m} d(v_i, v_j)}\alpha g \qquad (1)$$

where g is the gradient sign (either 1 or -1), α is the step size, and $d(v_i, v_j)$ is the Euclidean distance between the variables v_i and v_j.

3. SYA SYSTEM

Knowledge base construction (KBC) has been an active area of research over the last two decades with several system prototypes coming from academia and industry, along with many important applications, e.g., web search, digital libraries, and health care. Most recently, KBC systems employed the idea of MLN to associate each extracted relation with a probability of how confident is the system that this relation is factual. DeepDive [25], an MLN-based system, has emerged as one of the most popular probabilistic KBC systems, applied in different domains (e.g., law enforcement [10], geology [29], and paleontology [17]).



(a) Sya Accuracy (b) TurboReg Latency

Figure 3: Initial Experiments.

Unfortunately, DeepDive does not fully utilize the underlying spatial information, which results in less accuracy in the factual scores. This is because of two reasons: (1) DeepDive treats any predicate in the knowledge base inference rules as a boolean function, which yields either true or false (i.e., satisfied or not). So, although one can define spatial predicates in DeepDive, internally DeepDive and its inference engine do not do anything special for any spatial predicate. (2) DeepDive estimates the factual scores of extracted relations only based on how much support for these relations in training data. However, in case spatial information exists, the relative distance between entities participating in the extracted relations should be considered as well.

To overcome the above limitations in DeepDive, we proposed Sya [21]; a spatial MLN-based KBC system, built using our SMLN framework. We re-implemented the existing grounding, inference and learning modules in DeepDive using the corresponding modules in SMLN. We initially evaluated Sya through building two real spatial knowledge bases about: (1) the water quality in Texas [27], namely, GWDB, and (2) the air pollution in New York city [15], namely, NYCCAS. Figure 3(a) shows the quality (i.e., accuracy) for both Sya and DeepDive, measured by the F1-score, when building these two knowledge bases. Sya has an improvement of 120% and 27% over DeepDive in GWDB and NYCCAS, respectively.

4. TURBOREG SYSTEM

Autologistic regression [9] is an important statistical tool for predicting spatial phenomena. Unlike standard logistic regression that assumes predictions of spatial phenomena over neighbouring locations are completely independent of each other, autologistic regression takes into account the spatial dependence between neighbouring locations while building and running the prediction model (i.e., neighbouring locations tend to systematically affect each other). Myriad applications require the autologistic regression model to be built over large spatial data. However, existing methods for autologistic regression (e.g., see [9]) are prohibitively computationally expensive for large grid data, e.g., fine-grained satellite images, and large spatial epidemiology datasets. It could take about week to infer the model parameters using training data of only few gigabytes [9].

To solve this issue, we introduced TurboReg [20]; a scalable framework for building autologistic models with large number of prediction and predictor variables. In TurboReg, we employed the inference and learning modules of SMLN to estimate the autologistic model parameters in an accurate and efficient manner. Basically, TurboReg provides an equivalent first-order logic [5] representation to dependency relations among neighbours in autologistic models. In particular, TurboReg transforms each neighbouring dependency relation into a predicate with bitwise-AND operation on all variables involved in this relation. For example, a binary dependency relation between neighbouring variables C_1, and C_2 is transformed to $C_1 \wedge C_2$. This simple logical transformation allows non experts to express the dependency relations within autologistic models in a simple way without needing to specify complex details.

We experimentally evaluated TurboReg using a real dataset of the daily distribution of bird species [26], and compared its scalability performance to a state-of-the-art method, namely ngspatial [8]. Figure 3(b) shows the running time for both systems while building an autologistic model over grid sizes ranging from 250 to $84k$ cells. TurboReg has at least three orders of magnitude reduction in the running time over ngspatial, while preserving the same accuracy in estimating the model parameters.

5. FLASH FRAMEWORK

Same as MLN made it possible for data scientists and developers to embrace the difficulty of deploying machine learning techniques, we aim to use our proposed SMLN framework as a backbone infrastructure to support long lasting spatial analysis techniques that lack scalability as well as suffer from difficulty of deployment (e.g., [3, 4]).

To that end, we proposed Flash [22]; a framework for *generic* and *scalable* spatial data analysis. Flash focuses on building a major class of spatial analysis techniques, called *spatial probabilistic graphical modelling* (SPGM) (e.g., [3, 4, 6]), which uses probability distributions and graphical representations to describe spatial phenomena and make predictions about them. In Flash, we built a generic transformation module that is responsible to generate a set of *weighted* logical rules representing any SPGM input. The generated rules are then executed normally through the SMLN framework, where the weights of these rules represent the parameters of the original SPGM and will be inferred using the SMLN inference and learning modules. Currently, Flash supports transformation for three spatial graphical models; spatial Markov random fields [4], spatial hidden Markov models [6] and spatial Bayesian networks [3].

6. CONCLUSION

In this paper, we introduce SMLN; a framework that injects the spatial awareness inside the core functionality of Markov Logic Networks (MLN). SMLN equips the MLN framework with a spatial high-level language, and spatially-optimized grounding, inference and learning modules. Data scientists and developers can exploit SMLN to build numerous scalable and efficient spatial machine learning and analysis applications. We showed three on-top applications; Sya, a system for spatial probabilistic knowledge base construction, TurboReg, a framework for scaling up spatial autologistic regression models, and Flash, a framework for scalable spatial data analysis.

7. REFERENCES

[1] W. G. Aref and H. Samet. Efficient Processing of Window Queries in the Pyramid Data Structure. In *PODS*, 1990.

[2] A. Auchincloss et al. A Review of Spatial Methods in Epidemiology. *Annual Review of Public Health*, 2012.

[3] bnspatial: Spatial Bayesian Networks, 2019. `cran.r-project.org/web/packages/bnspatial`.

[4] gamlss.spatial: Gaussian Markov Random Fields, 2019. `cran.r-project.org/web/packages/gamlss.spatial`.

[5] M. Genesereth and N. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers, 1987.

[6] P. J. Green and S. Richardson. Hidden Markov Models and Disease Mapping. *JASA*, 2002.

[7] A. Guttman. R-trees: A Dynamic Index Structure for Spatial Searching. *SIGMOD Record*, pages 47–57, 1984.

[8] J. Hughes. ngspatial: A Package for Fitting the Centered Autologistic and Sparse Spatial Generalized Linear Mixed Models for Areal Data. *The R Journal*, 2014.

[9] J. Hughes, M. Haran, and P. C. Caragea. Autologistic Models for Binary Data on a Lattice. *Environmetrics*, 2011.

[10] Human Trafficking: Forbes. `http://www.forbes.com/sites/thomasbrewster/2015/04/17/darpa-nasa-and-partners-show-off-memex/`.

[11] M. Kaiser et al. Goodness of Fit Tests for a Class of Markov Random Field Models. *The Annals of Statistics*, 2012.

[12] K. Li et al. In-database Batch and Query-time Inference over Probabilistic Graphical Models using UDA-GIST. *VLDB Journal*, 2017.

[13] G. Y. Lu and D. W. Wong. An Adaptive Inverse-distance Weighting Spatial Interpolation Technique. *Journal of Computer and GeoSciences*, 34(9):1044–1055, Sept. 2008.

[14] F. Niu et al. Tuffy: Scaling Up Statistical Inference in Markov Logic Networks Using an RDBMS. In *VLDB*, 2011.

[15] NYC OpenData. `https://data.cityofnewyork.us/Environment/NYCCAS-Air-Pollution-Rasters/q68s-8qxv`.

[16] Open Geospatial Consortium. `http://www.opengeospatial.org/`.

[17] S. E. Peters, C. Zhang, M. Livny, and C. Ré. A Machine Reading System for Assembling Synthetic Paleontological Databases. *PLoS One*, 9(12), 2014.

[18] T. Rekatsinas et al. HoloClean: Holistic Data Repairs with Probabilistic Inference. *VLDB Journal*, 2017.

[19] M. Richardson and P. M. Domingos. Markov Logic Networks. *Machine Learning*, 2006.

[20] I. Sabek, M. Musleh, and M. Mokbel. TurboReg: A Framework for Scaling Up Spatial Logistic Regression Models. In *SIGSPATIAL*, pages 129–138, 2018.

[21] I. Sabek, M. Musleh, and M. F. Mokbel. A Demonstration of Sya: A Spatial Probabilistic Knowledge Base Construction System. In *SIGMOD*, pages 1689–1692, 2018.

[22] I. Sabek, M. Musleh, and M. F. Mokbel. Flash in Action: Scalable Spatial Data Analysis Using Markov Logic Networks. In *VLDB*, 2019.

[23] N. A. Sakhanenko and D. J. Galas. Markov Logic Networks in the Analysis of Genetic Data. *Journal of Computational Biology*, 17(11):1491–1508, Nov. 2010.

[24] J. Sankaranarayanan et al. TwitterStand: News in Tweets. In *SIGSPATIAL*, 2009.

[25] J. Shin et al. Incremental Knowledge Base Construction Using DeepDive. In *PVLDB*, pages 1310–1321, 2015.

[26] B. Sullivan et al. eBird: A Citizen-based Bird Observation Network in the Biological Sciences. *Biological Conservation*, 2009.

[27] Texas Ground Water Database. `http://www.twdb.texas.gov/groundwater/data/`.

[28] M. Wick et al. Scalable Probabilistic Databases with Factor Graphs and MCMC. In *PVLDB*, 2010.

[29] C. Zhang et al. GeoDeepDive: Statistical Inference Using Familiar Data-processing Languages. In *SIGMOD*, 2013.

[30] C. Zhang and C. Ré. Towards High-throughput Gibbs Sampling at Scale: A Study Across Storage Managers. In *SIGMOD*, pages 397–408, 2013.

[31] M. Zinkevich et al. Parallelized Stochastic Gradient Descent. In *NIPS*, pages 2595–2603, 2010.