Universität des Saarlandes
Max-Planck-Institut für Informatik
AG5

Learning Rules With Numerical Constants in Large Uncertain Knowledge Bases

Masterarbeit im Fach Informatik
Master's Thesis in Computer Science
von / by

André de Oliveira Melo

angefertigt unter der Leitung von / supervised by

Prof. Dr. Gerhard Weikum

betreut von / advised by

Dr. Martin Theobald

begutachtet von / reviewers

Dr. Max Mustermann
Prof. Dr. Gerhard Weikum

November / November 2012

Hilfsmittelerklärung

Hiermit versichere ich, die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben.

Non-plagiarism Statement

Hereby I confirm that this thesis is my own work and that I have documented all sources used.

Saarbrücken, den 19. November 2012,

(André de Oliveira Melo)

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlichtwird.

Declaration of Consent

Herewith I agree that my thesis will be made available through the library of the Computer Science Department, Saarland University.

Saarbrücken, den 19. November 2012,

(André de Oliveira Melo)

To my father Cicero, my mother Marlene and my sister Carolina

- Andre

Abstract

With millions of articles in multiple languages, Wikipedia has become the de-facto source of reference on the Internet today. Each article on Wikipedia contains encyclopedic information about various topics (people, events, inventions, etc.) and implicitly represents an entity. Extracting the most important facts about such entity will help users to find desired information more quickly and effectively. However, this task is challenging due to the incomplete and noisy nature of Wikipedia articles. This calls for a mechanism to detect and summarize the most important information about an entity on Wikipedia.

This thesis proposes and implements CATE (**C**ontext-**A**ware **T**imeline for **E**ntity Exploration), a framework that utilizes Wikipedia to summarize and visualize the important aspects of entities in a timeline fashion. Such a system will help users to draw quickly an informative picture of an entity (e.g. life of a person, or evolution of a research topic, etc.). The novelty of CATE lies in seeing the entity in different contexts, synchronous with contemporaneous events. In addition, CATE puts the entity in a relationship with other entities, and thus offers a broader portrait about it. In order to efficiently query and visualize the events related to the entity, a number of techniques have been developed, combining information extraction and information retrieval with a novel ranking model. The thesis also discusses several experiments and evaluation results to show the effectiveness of the methods proposed.

Acknowledgements

Firstly, I would like to thank my advisor Dr. Martin Theobald, for his invaluable guidance. I feel deeply grateful for his technical assistance and motivational encouragement.

A special note of thanks to Prof. Gerhard Weikum for giving me the opportunity to pursue this thesis at Infomation and Database Systems department under his supervision. It was a very enriching and pleasant experience to write my Master Thesis here.

Contents

Chapter 1

Introduction

In the last years, the volume of semantic data available, in particular RDF, has dramatically increased. Initiatives like the W3C Semantic Web, which provides a common standard that allows data to be shared and reused across different applications, and the Linked Open Data, which provides linkages between different datasets that were not originally interconnected, have great contribution in such development. Moreover, advances in information extraction have also made strong contribution, by crawling multiple non-structured resources in the Web and extracting RDF facts.

Nevertheless, information extraction still has its limitations and many of sources might contain contradictory or uncertain information. Therefore, many of the extracted datasets suffer from incompleteness, noise and uncertainty.

In order to reduce such problems, one can apply to the knowledge base a set of inference rules that describes its domain. With that, it's possible to resolve contradictions or strengthen or weaken their confidence values. It's also possible to derive new facts that are originally not existent due to incompleteness. Such inference rules can be of two types:

1. *Hard Rules*: Consistency constraints which represent which might represent functional dependencies, functional or inverse-functional properties of predicates or Mutual exclusion. For example:

    - $marriedTo(x, y) \leftarrow marriedTo(y, x), (x \neq y)$
    - $grandChildOf(x, y) \leftarrow childOf(x, z), childOf(z, y)$
    - $parentOf(x, y) \leftarrow childOf(y, x)$
    - $(z = y) \leftarrow wasBornIn(x, z), wasBornIn(x, y)$

2. *Soft Rules*: Weighted rules that frequently, but not always hold in the real world. As they might also produce incorrect information, each rule itself must have a confidence value which should be applied to derived facts, for example married people live in the same place as their partner has confidence 0.8:

$$livesIn(x, y) \leftarrow marriedTo(x, z) livesIn(z, y) \text{ [0.8]}$$

So, if we have an incomplete knowledge base, which lacks information about where *Michelle Obama*l lives, but we know that she's married to *Barack Obama* and he lives in *Washington, D.C.*, both with confidence 1, we could then apply this soft rule to derive the fact *livesIn(MichelleObama, WashingtonDC)* with confidence 0.9.

Such rules are rarely known beforehand, or are too expensive to be manually extracted. Nevertheless, the data itself can be used to mine these rules using *Inductive Logic Programming (ILP)*.

ILP is a well-established framework for inductively learning relational descriptions (in the form of logic programs) from examples and background knowledge. Given a logical database of facts, an ILP system will generate hypothesis in a pre-determined order and test them against the examples. However in a large knowledge base, ILP becomes too expensive as the search space grows combinatorially with the knowledge base size and the larger the number of examples, the more expensive it is to test each of the hypothesis.

rules with constants might be really interesting.

Moreover, testing hypothesis with constants increases the search space dramatically, making it unfeasible to test all possible hypothesis with constants in a large knowledge base. In such case, it's necessary to arbitrarily reduce the search by restricting the set of constants to be included in ...

data mining, rule mining, datalog rules

talk a bit about ilp

1.1 Motivation

Given the huge size of search space and the great interestingness of rules with constants, we need to smartly prune constants or combinations of constants that , learning datalog rules can be already extremely costly.

Numerical constants are a special case, and they need to be treated differently. Depending on the numerical attribute domain, in case of a continuous real number domain for example, setting a numerical constant as an individual value will very likely have a very low support and also that would result and extremely large number of possible constants. Therefore, we split the attribute's domain into k buckets and then check if any of the buckets present any gain in comparison with its correspondent numerical constant-free hypothesis.

For example if we test the hypothesis and we find support=100 and confidence=0.4:

$$isMarriedTo(x, y) \leftarrow hasAge(x, z)$$

and then we split z into three buckets:

- $k = 1 : z \in [0, 20]$
- $k = 2 : z \in (20, 40]$
- $k = 3 : z \in (40, \infty]$

we then test the hypothesis for each of the three buckets and we obtain

- $isMarriedTo(x, y) \leftarrow hasAge(x, z), z \in [0, 20]$
  support=40, confidence=0.1
- $isMarriedTo(x, y) \leftarrow hasAge(x, z), z \in (20, 40]$
  support=40, confidence=0.5
- $isMarriedTo(x, y) \leftarrow hasAge(x, z), z \in (40, \infty]$
  support=20, confidence=0.8

as we see, for k=2 and k=3, the hypothesis we have significant gain by specifying numerical constants. Adding a relation to the body might produce totally different confidence support and confidence distributions along the buckets. For example, if we add the relation hasChild(x,a), we could obtain other interesting rules:

- $isMarriedTo(x, y) \leftarrow hasAge(x, z)hasChild(x, a)$
  support=50, confidence=0.625
- $isMarriedTo(x, y) \leftarrow hasAge(x, z)hasChild(x, a), z \in [0, 20]$
  support=2, confidence=0.5
- $isMarriedTo(x, y) \leftarrow hasAge(x, z)hasChild(x, a), z \in (20, 40]$
  support=30, confidence=0.7

- $isMarriedTo(x, y) \leftarrow hasAge(x, z)hasChild(x, a), z \in (40, \infty]$
  support=18, confidence=0.9

adding some relations might not bring any gain or even loss in confidence, but when bucketing per age, present a different distribution,

nevertheless, adding some relations might not generate any interesting rules, like

d

1.2 Contributions

In this Thesis, we propose a pre-processing step to build a graph we call Correlation Latticefor each numerical property. In each graph, that has a numerical property as root, we first query the frequency distribution on the numerical attribute, then split them in k buckets. Then we pick a set of c categorical properties that can be joined with the root, and analyze how the distribution of sub-population created by joining them with the root is affected. Afterwards we try to combine each of the categories and see if they still produce interesting sub-populations, like in frequent set mining.

We also evaluate different heuristics and interestingness measures.

In a hypothesis containing a numerical attribute in the body, we can obtain a support and confidence value for each of the buckets, and

With that, during the ILP algorithm, once we add one of the root properties, we can then search for the most interesting categorical properties that could result in different accuracy distributions. For every categorical property we can also suggest the most interesting constants and other categorical properties to be combined in a subcategory of both.

1.3 Outline

The remainder of this thesis is structured as follows. In Chapter **??**, we provide technical background on MapReduce and BigTable. In Chapter **??**, we present a summary of previous work in the areas of duplicate and near-duplicate detection, information retrieval on web archives, and MapReduce applications in graph processing. Following that, we state our problem and describe solutions in Chapter **??**. In Chapter **??**, we describe an implementation of our solution using the MapReduce framework. In Chapter **??**, we present our experimental results. We conclude this thesis and outline directions of future research in Chapter **??**.

Chapter 2

Related Work

2.1 Logic Programming

2.2 Inductive Logic Programming

2.3 Mining Opimized Rules for Numberic Attributes

2.4 Minimum Description Length

2.5 Semamtic Web

2.6 Linked Open Data

Chapter 3

Correlation Lattice

Chapter 4

Algorithmic Framework

4.1 Preprocessing

In this section, we will present the preprocessing steps required by our proposed algorithm. It basically consists of building a joinable repations map for each of the four join patterns, according to relations domain and range types as well as support threshold. Afterwards, we search the available categorical properties for each numerical relation that will be used in the Influence Graphs. At last we describe an Influence Graph and the algorithm to build it.

4.1.1 Relation Preprocessing

In this step, we focus on creating for each of the four join patterns between two relations:

- Argument 1 on Argument 1: e.g. hasIncome(**x**,y)hasAge(**x**,z)
- Argument 1 on Argument 2: e.g. hasIncome(**x**,y)isMarriedTo(z,**x**)
- Argument 2 on Argument 1: e.g. livesIn(y,**x**)isLocatedIn(**x**,z)
- Argument 2 on Argument 2: e.g. livesIn(y,**x**)wasBornIn(z,**x**)

4.1.1.1 Exploiting Relation Range and Domain Types

A knowledge base is expected to have an ontology defining the structure of the stored data (the types of entitties and their relationships). Additionally, every relation's range (type of 1st argument) and domain (type of 2nd argument) should be defined. These information can help us identify the allowed joining relations for each join pattern.

For every possible pair of relations,

The algorithm is shown in the pseudo-code bellow:

4.1.1.2 Exploiting Support Monotonicity

As seen in (???), support is the only monotonically decreasing measure in top-down ILP. So we know that by adding any literals to the hypothesis, we can only get a smaller or equal support. Therefore, for each pair of joinable realtions in each of the join patterns, we can query the knowledge base and check whether they reach the minimum support threshold.

Thus, if any pair of relations doesn't reach the minimum support for a given join pattern, we know that any hypothesis containing such join will therefore fail the support test as well, so we don't need to test such hypothesis in the core ILP algorithm.

The relation preprocessing will result in 4 maps, one for each join pattern. Each map will a relation as key and a set of joinable relations as value. The refinement step at the ILP algorithm, will then access this map when choosing a new literal to be added.

4.1.2 Influence Graph

4.2 ILP Core Algorithm

Data: this text
Result: how to write algorithm with LaTeX2e
initialization;
while *not at end of this document* **do**
    read current;
    if *understand* **then**
        go to next section;
        current section becomes this one;
    else
        go back to the beginning of current section;
    end
end

Algorithm 1: How to write algorithms

Algorithm 2: Checks valid join pairs for a given join patterns

List of Figures

List of Tables

Bibliography