Interactive Rule Mining in Knowledge Bases

Luis Galárraga
Télécom ParisTech
Paris, France
galarrag@enst.fr

ABSTRACT

L'extraction de règles d'association dans une base de connaissance consiste en la découverte de façon automatique de règles logiques à partir de faits connus. Cette tâche reste cependant un défi pour plusieurs raisons. D'abord, les bases de connaissances ne contiennent pas d'informations négatives, c'est-à-dire, les méthodes d'extraction de règles ne disposent pas de contre-exemples. En outre, les bases de connaissance existantes sont volumineuses avec des millions de faits ; cela accroit l'espace de recherche de manière considérable. Les systèmes d'extraction de règles actuels réduisent l'espace de recherche par une exclusion systématique des candidats peu prometteurs pendant le processus de construction des règles.

Dans cet article, nous proposons un processus d'extraction de règles transparent à l'utilisateur au travers un modèle de démonstration interactif.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Rule Mining, Knowledge Bases

1. INTRODUCTION

A knowledge base (KB) is a collection of machine-readable information designed to model a certain domain of knowledge. Initiatives such as YAGO [19], DBpedia [2], Wikidata[1], and the Knowledge Vault [7], among others, have constructed large-scale KBs about people, places, organizations, etc., by applying Information Extraction (IE) techniques on the Web [8, 19, 6, 3]. KBs are also prominent in the Life Sciences, with projects such as BioRDF[2] or UniProt[3]. KBs are crucial when computers have to process natural language text in a semantic fashion. The Knowledge Vault, for instance, allows the Google search engine to identify concepts and proper names in the words of queries. This facilitates the retrieval of answers about real-world entities instead of pages matching the words in the query.

KBs can be mined for patterns and rules. For instance, we could learn that Grammy awardees frequently play guitar. We could also find out that usually, people live in the same place as their spouses. If we resort to a datalog-like notation of facts, these examples can be formulated as logical rules:

$$musicalRole(x, Guitar) \Leftarrow wonPrize(x, Grammy) \quad (1)$$

$$livesIn(x, y) \Leftarrow isMarriedTo(z, x) \wedge livesIn(z, y) \quad (2)$$

These rules provide not just insight by themselves, they can also be used to predict information or propose corrections to the data. For example, assume that the KB knows *livesIn(Barack Obama, Washington)* and *marriedTo(Barack Obama, Michelle)* but does not know the place of residence of *Michelle*. Then we can use the second rule to deduce *livesIn(Michelle, Washington)*, that is, rules can be used to deduce new information [10]. We can also use the rules to correct erronous facts. If, say, the KB states that Michelle lives in a different city, then this statement can be flagged as suspicious. Other applications of rules include data maintenance tasks such as schema inference [20], KB alignment [11], or KB canonicalization [9].

Rule mining on KBs is a challenging endeavor for multiple reasons. First, today's KBs scale up to millions of facts and entities, and thus the search space for rule mining is huge. Second, Web-extracted KBs often contain gaps and noise that hinder the discovery of frequent and interesting patterns. Third, while mining logical rules on structured data has been widely studied by the Inductive Logic Programming (ILP) community, traditional ILP methods require explicit counter-examples. KBs, in contrast, operate under the *Open World Assumption* (OWA) and do not represent negative information.

AMIE [10] is a system that tackles the aforementioned challenges. The system is designed to mine rules on large KBs under the OWA. This is possible thanks to a novel confidence metric that gauges the quality of a rule even in the absence of counter-examples. Furthermore, AMIE is very efficient, i.e., it can mine rules on KBs with millions of facts in a matter of minutes. The key of AMIE's scalability is its tailored data-storage implementation and its wide range of pruning techniques, which cut out unpromising rules early on in the mining. These techniques include thresholding on

[1] `http://www.wikidata.org`
[2] `http://bio2rdf.org/`
[3] `http://www.uniprot.org/`

BDA 14 octobre 2014, Grenoble-Autrans, France.

support and confidence values, exploiting monotonicity, and a lookahead on the next step.

To the user, however, the rule mining process appears as a black box: In its default setting, AMIE does not take any parameters or language bias as input, and so the user just points AMIE to the KB file and lets her run. AMIE takes some minutes to load the data and compute the rules, and then outputs the result. Thus, the user does not actually understand how these rules are mined or why certain rules were not mined.

With this demo, we aim to shed light on the inner workings of AMIE. We want to show the user why certain rules are mined, for what reason certain others are discarded, and how AMIE maneuvers in a search space that appears near infinite to the user. Our demo allows the user to "join" AMIE live in the mining process, and to see and influence the decisions that AMIE makes.

2. RELATED WORK

Several systems have been developed for rule mining [21, 17, 4, 12, 15, 5, 14, 16] on KBs. None of them, however, is explicitly designed to mine rules on large and potentially incomplete KBs under the OWA as AMIE. WARMR [12] is a system that reconciles ILP and association rule mining. In the spirit of traditional association rule mining, WARMR operates under the Closed World Assumption (CWA), that is, if a rule makes a prediction that is not present in the KB, the system counts it as a counter-example for the rule. ALEPH [14] is another ILP system, which offers a suite of different metrics to evaluate the quality of rules. One of them is the positives-only learning function, which does not require explicit counter-examples, but evaluates rules based on their length, their number of positive examples and the number of randomly generated facts covered by the rule. In [10] it is shown that AMIE's confidence metric outperforms ALEPH's positives-only learning function in terms of precision, when the rules are used for data inference. Quickfoil [21] is a traditional ILP system, which unlike WARMR or ALEPH, scales to millions of facts. Quickfoil, on the other hand, requires the user to provide explicit negative information for rule induction.

There have been also some work on unveiling the internals of rule mining to the users, and showing its applications. Crowdminer [1] is an interactive system that applies association rule mining techniques on data collected from the crowd. The system combines a proactive crowdsourcing approach to ask the users the most *benefitial* questions (those that maximize the knowledge gain) about their well-being practices such sports or dietary habits. The system allows the user to navigate through the most significant association rules and their metrics, e.g., for Crowdminer, a transaction is a bag of terms and rule looks like $\{coffee\} \Leftarrow \{headache\}$. SPIRE [13] is a tool targeted to data analysts that allows for intuitive visualization of the rule mining search space. One particularity of SPIRE is its support for negative conditions in rules, e.g., in our context rules such as $\neg deathPlace(x, z) \Leftarrow is(x, LivingPerson)$. The system can help data scientists identify the rules and correlations that are relevant to certain application or scenario, in particular when rules are numerous.

3. PRELIMINARIES

Logical rules are built on *atoms*. An atom is of the form $r(x, y)$, where r is a relation x and y are either entities (such as *Barack Obama*) or variables (such as a). At least one of the arguments has to be a variable. A *Horn rule* is an expression of the form

$$r(x, y) \Leftarrow \boldsymbol{B}$$

where $r(x, y)$ is the *head* or *conclusion* of the rule, and $\boldsymbol{B}$ is a sequence of atoms $r_1(x_1, y_1), \ldots, r_m(x_m, y_m)$ called the *body*. Two atoms are connected if they share at least one variable. A rule is *connected* if every atom in the rule is transitively connected to every other atom. A rule is *closed* if every variable appears in at least two different atoms.

Rules encode patterns that occur frequently in the data. This notion of frequency is measured by the *support* of the rule. Support is calculated as

$$supp(r(x, y) \Leftarrow \boldsymbol{B}) := \#(x, y) : \exists z_1, ..., z_m : r(x, y) \wedge \boldsymbol{B}$$

Here, $\#(x, y) : \Phi$ are the number of pairs (x, y) that fulfill the condition Φ in the KB. A normalized version of the support is called the *head coverage* and is defined as:

$$hc(r(x, y) \Leftarrow \boldsymbol{B}) := \frac{supp(r(x, y) \Leftarrow \boldsymbol{B})}{\#(x, y) : r(x, y)}$$

The head coverage normalizes the support of the rule over the size of the head relation. This is convenient for KBs with small relations that would be ignored even for reasonable support thresholds. Support and head coverage are monotonic metrics, i.e., the addition of new atoms to a rule will never increase its support. This property is crucial for the AMIE algorithm. Support and head coverage are measures of statistical evidence. However, they do not gauge the *precision* of the rule. This dimension is taken into account by the standard confidence metric:

$$stdconf(r(x, y) \Leftarrow \boldsymbol{B}) := \frac{supp(r(x, y) \Leftarrow \boldsymbol{B})}{\#(x, y) : \exists z_1, ..., z_m : \boldsymbol{B}}$$

The standard confidence normalizes the support of the rule by the number of bindings of the head variables that match the antecedent of the rule. In other words, it measures the ratio of correct conclusions drawn from the rule. The standard confidence operates under the Closed World Assumption (CWA): It counts as counter-evidence all the conclusions of the rule that do not appear in the KB. However, under the OWA, conclusions may be true even if they are not in the KB. Hence, AMIE proposes an alternative confidence metric, as we shall see.

4. THE AMIE SYSTEM

4.1 Functions

A *function* is a relation that maps each subject to at most one object, such as *place of birth*. A relation is an *inverse function*, if its inverse is a function. Due to the noise in Web-extracted KBs, it is rare to find conceptually functional relations that comply strictly with these definitions. Therefore, [18] defines a functionality score for relations:

$$fun(r) := \frac{\#x : \exists y : r(x, y)}{\#(x, y) : r(x, y)}$$

The functionality is 1 for functional relations, and decreases towards 0 for less functional relations. The inverse functionality of a relation is defined as the functionality of its inverse relation $ifun(r) := fun(r^{-1})$. AMIE assumes that relations are more functional than inverse functional ($fun(r) \geq ifun(r)$). If it is not the case, a relation r can always be replaced by its inverse relation r^{-1}. This is called the FUN-property of KBs in [10].

4.2 PCA Confidence

The Partial Completeness Assumption (PCA) [10] assumes that if a KB knows *some r-values* for an entity, then it knows *all its r values* for that entity.[4] For example, if the KB knows 4 official languages for Switzerland, then the PCA assumes than Switzerland has no further official languages. If a rule predicts a new language, the PCA labels the conclusion as false. If, in constrast no official language for Switzerland is known, then the PCA will not assume anything, i.e., it will label any conclusion for the language of Switzerland as unknown. The CWA, in contrast, would label both conclusions as false.

This means that the PCA admits that the KB can be incomplete. Therefore, it is more suitable for rule mining under the OWA. Under the PCA, we compute the confidence of a rule as follows:

$$pcaconf(r(x,y) \Leftarrow \boldsymbol{B}) := \frac{supp(r(x,y) \Leftarrow \boldsymbol{B})}{\#(x,y) : \exists z_1, \ldots, z_k, y' : r(x,y') \wedge \boldsymbol{B}}$$

The PCA confidence normalizes the support of the rule over the number of positive cases and the cases that the PCA assumes to be false. Experiments [10] suggest that the PCA confidence is more effective than the standard confidence at identifying productive rules, i.e., rules that produce many correct conclusions beyond the ones known to the KB.

4.3 Algorithm

The AMIE algorithm explores the search space of closed Horn rules by a breadth-first search. The algorithm starts with a queue that contains only the empty rule (with no head and no body). In each iteration, the algorithm dequeues a rule and applies one of the following mining operators:

- **Add Dangling Atom** ($\mathcal{O}_\mathcal{D}$)
  This operator adds a new atom to a rule. The new atom uses a fresh variable for one of its two arguments. The other argument (variable or entity) is shared with the rule, i.e., it occurs in some other atom of the rule.

- **Add Instantiated Atom** ($\mathcal{O}_\mathcal{I}$)
  This operator adds a new atom that uses an entity for one of its arguments, and shares the other argument (variable or entity) with the rule.

- **Add Closing Atom** ($\mathcal{O}_\mathcal{C}$)
  This operator adds a new atom that shares both of its arguments with the rule.

The new, refined, rules are then added to the queue. This process is run in multiple threads until the queue is empty. AMIE relies on a tailored in-memory triple store, which is optimized for the kind of queries that the operators generate. This component in combination with the pruning (s.b.) allows AMIE to mine closed Horn rules on KBs three orders of magnitude faster than other state of the art systems such as WARMR [12] or ALEPH [15].

[4]The PCA was later used as the *local closed world assumption* in [7].

4.4 Pruning

If run naively, the AMIE algorithm would enumerate the entire search space. Thus, it would not finish in reasonable time. Therefore, AMIE limits the search as follows:

1. By default, AMIE mines rules only up to 3 atoms. The user can choose to increase this threshold, but experiments [10] show that longer rules capture only very obscure correlations in practice.

2. If a rule has head coverage below 1%, AMIE considers it too insignificant, and filters it out. However, the user can also provide a different threshold value or prune on absolute support instead.

Since support, head coverage, and the length of the rule are monotonic metrics, AMIE can safely abandon any further refinement of such rules as well.

In addition, AMIE uses the following pruning techniques:

1. If the rule is closed and has a PCA confidence of 1.0, then no refinement can make the rule better, because confidence cannot increase and support can only decrease. Hence, the rule is output and removed from the queue. We call these rules *perfect rules*.

2. If the new rule has a lower confidence than the rule that it is derived from, then the rule is just silently enqueued for further refinement, but not output. This is because it will have lower confidence *and* lower support than the previous rule. This technique is called the *skyline technique* and was designed to avoid overspecifications of the same logical rule.

3. If the maximum permitted length of a rule is n, then AMIE will not apply the operator $\mathcal{O}_\mathcal{D}$ at length $n-1$. This is because the operator adds a new variable, and thus the rule cannot be closed within the limit of n atoms. This technique is called the *lookahead*.

4. When AMIE adds a new instantiated atom, it will exclude atoms where the variable of this atom can bind to only one entity in the KB. This is because these cases induce a *quasi-binding* of the free variable, meaning that there is a shorter equivalent rule. For example, it does not make sense to add the atom *hasOfficialLanguage(x,Romansh)* to a rule, because x can bind only to *Switzerland*. Thus, there is a shorter equivalent rule, namely the one where the atom is not added and x is replaced by *Switzerland*.

5. DEMO

5.1 Overview

Our demo lets the user understand the heuristics that govern AMIE's search strategy by allowing the user to navigate through the search space together with AMIE. For this purpose, our demo shows the rule construction process, starting from the empty rule. At each step, the user can add a new

Figure 1: Mining a closed Horn rule

atom $r(x, y)$. This involves two steps: (1) choosing the relation r and (2) choosing the arguments x, y. The second step offers various permutations of arguments: Each argument can be a variable (either known or fresh) or an entity (chosen from the set of entities that lead to the highest support). For instance, imagine the user wants to follow the path to mine the rule

$$livesIn(a, b) \Leftarrow isMarriedTo(a, f) \wedge livesIn(f, b)$$

The demo starts with the empty rule "... $\Leftarrow$ ..." and asks the user to select a relation for the head atom. The system shows as choices a list of relations ranked by size. Once the user picks a relation (i.e., *livesIn* in our example), the system applies the mining operators on the empty rule and reports all the refinements that have *livesIn* as relation, ranked by support. This includes, for example, the atom $lives(a, b)$ from the operator $\mathcal{O}_{\mathcal{D}}$, or the atom $livesIn(a, California)$ from the operator $\mathcal{O}_{\mathcal{I}}$. When applied the first time, the operator $\mathcal{O}_{\mathcal{I}}$ binds always the least functional variable of relations. In our example, it is more reasonable to predict the place of residence of a person, that all the people that live in a place (Section 4.2). Once the arguments are fixed, the atom is appended to the rule, and the user can choose the next atom. In the example, the user would select the relation $isMarriedTo$, followed by the arguments a, b, and then in a new atom the relation $livesIn$ (Figure 1). The user can also, at any point, decide to backtrack and to remove the last atom from the rule in order to explore a different path in the search space.

At each step in the process, the user has complete freedom to choose his options. However, the system also shows the metrics and pruning strategies that AMIE would apply. Our demo shows for every possible choice of atoms the support, the head coverage, the standard confidence, and the PCA confidence that this atom would achieve. Thus, the user can see which choice achieves the best metrics, and why – much like AMIE does during the mining process. For illustration, we also show positive examples, negative examples, and negative examples under the PCA for the new rule.

5.2 Implementation

Our demo is implemented as a client-server application that lets the user drive the AMIE algorithm step by step. We use the YAGO ontology [19] as KB. In order to guarantee reasonable response times in the interactive client-server setting, we created a sample of YAGO following the same procedure as in [10]: We took 10K random seed entities, and collected all the facts within a range of 3 hops from the seed entities, producing a sample of 35K facts. The server consists of a Java servlet that serves as interface to the AMIE codebase, namely the mining operators described in Section 4.3 and the in-memory triple store. The KB is loaded into memory only once when the servlet is initialized in the servlets container. The client side is a lightweight user-interface written in Javascript and HTML.

For the AMIE algorithm, it does not matter whether an unknown person or President Obama is a counter-example for a rule. For the user, in contrast, it is more illustrative to show prominent entities rather than unknown ones. Hence, we computed a relevance score for each entity e as:

$$relevance(e) := log(wikilength(e)) \times (incoming(e) + 1)$$

Here, $wikilength(e)$ is the length of the Wikipedia article of the entity (in bytes), and $incoming(e)$ is the number of Wikipedia articles linking to the article of e. Both numbers can be easily obtained from YAGO. We add 1 in the second term to guarantee that the score is a positive number. The relevance of a fact $r(x, y)$ is defined as the sum of the relevance scores of its arguments. This score is used to rank facts when displaying examples for rules, so that facts about prominent entities are preferred.

6. CONCLUSION

Rule Mining is a fascinating process, in which nuggets of insightful correlations are mined on millions of facts. Our demo allows the user to join AMIE in this process on a sample KB, and to discover how the search space can be cut down to a tractable size. Our demo is fully implemented, and available online at `http://luisgalarraga.de/amie-demo/`. Through the interaction with the demo participants, we hope not just to shed light on the inner workings of AMIE, but also to develop ideas for new pruning strategies together with the participants.

7. ACKNOWLEDGMENTS

This work is supported by the Chair "Machine Learning for Big Data" of Télécom ParisTech.

8. REFERENCES

[1] Y. Amsterdamer, Y. Grossman, T. Milo, and P. Senellart. Crowd miner: Mining association rules from the crowd. In *Proc. VLDB*, Riva del Garda, Italy, 2013. Demo paper.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A nucleus for a web of open data. In *ISWC*, 2007.

[3] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. H. Jr., and T. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

[4] L. Dehaspe and H. Toironen. Discovery of relational association rules. In *Relational Data Mining*. Springer-Verlag New York, Inc., 2000.

[5] L. Dehaspe and H. Toivonen. Discovery of frequent DATALOG patterns. *Data Min. Knowl. Discov.*, 3(1), Mar. 1999.

[6] L. Del Corro and R. Gemulla. Clausie: clause-based open information extraction. In *WWW*, 2013.

[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 *KDD*, 2014.

[8] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam. Open Information Extraction: the Second Generation. In *IJCAI*, 2011.

[9] L. Galárraga, G. Heitz, K. Murphy, and F. Suchanek. Canonicalizing Open Knowledge Bases. In *CIKM*, 2014.

[10] L. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek. Fast Rule Mining in Ontological Knowledge Bases with AMIE+. *VLDB Journal*, 2015.

[11] L. A. Galárraga, N. Preda, and F. M. Suchanek. Mining rules to align knowledge bases. In *AKBC*, 2013.

[12] B. Goethals and J. Van den Bussche. Relational Association Rules: Getting WARMER. In *Pattern Detection and Discovery*, volume 2447. Springer Berlin / Heidelberg, 2002.

[13] X. Lin, A. Mukherji, E. A. Rundensteiner, and M. O. Ward. Spire: Supporting parameter-driven interactive rule mining and exploration. *Proc. VLDB Endow.*, 2014.

[14] S. Muggleton. Inverse entailment and progol. *New Generation Comput.*, 13(3&4), 1995.

[15] S. Muggleton. Learning from positive data. In *ILP*. Springer-Verlag, 1997.

[16] V. Nebot and R. Berlanga. Finding association rules in semantic web data. *Knowl.-Based Syst.*, 25(1), 2012.

[17] S. Schoenmackers, O. Etzioni, D. S. Weld, and J. Davis. Learning first-order Horn clauses from web text. In *EMNLP*, 2010.

[18] F. M. Suchanek, S. Abiteboul, and P. Senellart. Paris: Probabilistic alignment of relations, instances, and schema. *PVLDB*, 5(3), 2011.

[19] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge. In *WWW*, 2007.

[20] J. Voelker and M. Niepert. Statistical schema induction. In *ESWC*, 2011.

[21] Q. Zeng, J. Patel, and D. Page. QuickFOIL: Scalable Inductive Logic Programming. In *VLDB*, 2014.