Efficient Subgroup Discovery using Spark

Livin Natious[1] and Pardeep Kumar Naik[2]

[1] Informatik III, Universität Bonn, Germany
livin.natious@gmail.com
[2] Informatik III, Universität Bonn, Germany
pardeepnaik@gmail.com

Abstract

This paper deals with semantic subgroup discovery, a new data mining paradigm that uses ontologies as background knowledge. This paradigm is based on SDM-SEGS which is a subgroup discovery system that was introduced to search for enriched gene sets. The new data mining paradigm is implemented in Scala using Apache Spark, an open-source cluster-computing framework. It generates accurate rule-set that satisfies all conditions of given subgroup. At first, we generate all possible rules for given subgroup and then accurate rules are filtered out using WRAcc (Weight Related Accuracy).

1 Introduction

Knowledge discovery in databases (KDD) refers to the interactive and iterative process of finding interesting patterns and model in data[1]. For knowledge discovery task, given data is pre-processed and then a data mining algorithm is applied to generate rules which are used in problem-solving or prediction tasks.

Data mining tasks includes classification, regression, clustering and association analysis, but not limited to only these tasks. In this paper, subgroup discovery task is discussed. The task of subgroup discovery was defined by Klsgen[2] and Wrobel[3] as follows: 'Given a population of individuals and a property of those individuals that we are interested in, find population subgroups that are statistically most interesting'. The rules generated by subgroup discovery are in the form of Class←Conditions, where condition is logical conjunction of ontology terms that satisfy a selected class of interest.

On the basis of literature on inductive logic programming (ILP) and relational data mining (RDM), one can say that, if the relation between the attribute values are taken into account, the performance of data mining methods can be improved significantly. In other words, domain knowledge can be used in knowledge discovery for providing additional benefits. Ontologies (as background knowledge) has not been exploited yet in the ILP and RDM literature. Ontologies can provide additional information to data mining algorithms.

As the semantic web is getting popular and various ontologies are available, the semantic data is growing day by day. The domain knowledge can be represented in a format (e.g. web ontology language (OWL) for ontologies and resource description framework (RDF) for other structured data) which can be reused in data mining algorithms.

RDF is a data model which can be represented in the form of subject-predicate-object. The data from multiple heterogeneous sources can be integrated into RDF data model. More and more data from public relational databases is converted into RDF as linked data, hence the data can be easily queried over multiple data repositories using semantic descriptors.

In data mining, a lot of empirical data is used but the domain knowledge is rarely exploited because usually it can not be used directly. Now the challenge is to use the domain knowledge of semantically annotated data into data mining and knowledge discovery tasks. Semantic data mining deals with these new challenges and approaches of mining the semantic data in data mining tasks.

Data mining methods can be improved by utilizing semantic descriptors in data and by providing relations between attribute values. The induced hypothesis can be formed using ontologies. During the construction of hypothesis, the generalizations can be more effective by using high level ontological concepts. Number of ontologies related to biomedicine, biology, sociology, finance etc. are growing day by day. Ontologies can be utilized as semantic descriptors for the data in these applications and in multiple other applications too.

In previous paper, author describes three systems for semantic data mining. First, SEGS[4] (search for enriched gene sets) is a special purpose subgroup discovery system for analyzing gene expression microarray data. The other two systems used for semantic subgroup discovery. These two systems are influenced from the SEGS success, but are domain-independent:

1. SDM-SEGS[4]: a domain-independent semantic subgroup discovery system based on SEGS.

2. SDM-Aleph[4]: a domain-independent semantic subgroup discovery system based on the ILP system Aleph.

In previous paper[4], the author explained SDM-SEGS in detail and pseudo-code was also provided for implementation of this system. According to author, the novelties of the paper[4] are, introduction to two new semantic subgroup discovery systems that are SDM-SEGS and SDM-Aleph, creation of a publicly available semantic data mining toolkit named SDM-Toolkit. Other contributions are, evaluation of SDM-SEGS and SDM-Aleph systems basis experiments and comparison with SEGS, providing the facts that SEGS and SDM-SEGS are suitable for biological and biomedical domain data analysis, whereas SDM-Aleph is more suitable for general purpose system.

In this paper, our main focus is on SDM-SEGS. In further sections, approach to implement SDM-SEGS is defined and then implementation of SDM-SEGS is defined too. In later section, evaluation of SDM-SEGS is also described.

2 Approach

The key idea is Semantic Data Mining(SDM), to use the background knowledge along with the dataset to generate all accurate rules associated with the interesting subgroup. Here, the focus is on new semantic subgroup discovery system SDM-SEGS(SDM - Search for Enriched Gene Sets), which is a generalized version of SEGS(Search for Enriched Gene Sets). SEGS was again proposed by Vavpetic[4] and Lavrac[4] as subgroup discovery system exclusively to search for enriched gene sets.

2.1 Semantic data mining task definition

Semantic subgroup discovery[6] was later extended to Semantic data mining in [5].
Semantic data mining can be defined as:

Given: Set of domain ontologies, and dataset with a mapping to ontology terms.

Find: Hypothesis (or a prediction model or set of rules/patterns) that uses ontology terms that represents the given dataset.

Liu [7] explains semantic data mining like: 'We propose to exploit the advances of the semantic web technologies to formally represent domain knowledge including structured collection of prior information, inference rules, knowledge enriched datasets etc, and thus develop frameworks for systematic incorporation of domain knowledge in an intelligent data mining environment. We call this technology the semantic data mining'. When it comes to SDM-SEGS this definition can be shortened and can be defined in a concise form:

Given: Set of ontologies(background knowledge), dataset containing training examples, and an example-to-ontology mapping which associates all terms in example to ontology terms

Find: Hypothesis (or a prediction model or set of rules/patterns) that uses ontology terms that represents the given dataset.

Semantic data mining is depicted in Figure [1]. Semantic subgroup discovery system SDM-SEGS(SDM - Search for Enriched Gene Sets) is described in the following section.

2.2 Existing SDM system SEGS

SEGS is a domain-specific systems that takes ontologies and hierarchies as background knowledge to generate enriched gene set hypothesis. Unlike previous works in gene st enrichment[8,9]. SEGS not only test existing gene set for differential expression but also describes the enriched gene sets in form of conjunction of ontology terms.

SEGS has got the following components in it, as described in previous section:

1. *domain knowledge* is an internal representation of the Gene Ontology (GO) and Krypto Encyclopedia of Genes and Genomes(KEGG);

2. *training data*, which contains list of ranked genes;

3. *example-to-ontology mapping*, maps each gene with a number of GO and KEGG concepts;

4. in addition a binary relation *interacts* is also used to model interaction between two genes

The rule construction in SDM-SEGS is similar to the rule construction that is used in SEGS. The only drawback with SEGS is that it is domain specific. SEGS can be only used to generate enriched gene set hypothesis given some standard gene onotologies from GO or KEGGS. On the other hand SDM-SEGS is a generalized version of SEGS, that is domain-independent but based on SEGS. The following section discusses about SDM-SEGS in detail.

2.3 SDM-SEGS

This section describes SDM-SEGS which is a generalized version of SEGS. Unlike SEGS, SDM-SEGS exploits ontologies from any domain and with a standard format (N-Triples or OWL). Since SDM-SEGS is based on SEGS, it exploits maximum of four ontologies.

In further sub-sections, we describe main parts of SDM-SEGS: the input data, the hypothesis language, the rule construction, rule selection and evaluation principles.

2.3.1 Input

The inputs to the SDM-SEGS system are:

1. *domain knowledge* in the format of n-triple or OWL ontology;
2. *training data*, which contains list of ranked examples;
3. *example-to-ontology mapping*, maps each example with number of concepts in ontology;

Here, we can use both class-labeled data as well as ranked examples. In the case of class-labeled data, user can specify target class. In ranked examples, user can specify a threshold value which splits given example set into two classes (positive and negative classes) based on rank. In both cases, it is a binary classification problem.

2.3.2 Hypothesis Language

Hypothesis language is described by Class(X)←Conditions, where Conditions are conjunction of ontology terms that satisfy a specific class of interest.

As an example Class(X)←scientist(X) Λ germany(X), where both scientist and germany denotes ontology terms. This rule denotes subgroup of people who are German scientist.

2.3.3 Rule Construction

By specifying more specific top concept, more specific rules can be generated by skipping general rules which are not interesting. Rule construction generates a set of rules that satisfies size constraint (minimum support and maximum number of terms in the rule). Rule construction Figure 1(a) is a top-down bounded exhaustive search algorithm. Rule construction traverses through all possible rules by taking one concept from each ontology. It starts with a rule Class(X)← , which covers all example set. Then, the algorithm tries to conjunctively add top concept of first ontology to the rule. If the new rule satisfy all the size constraints, the rule is added to the ruleset and recursively tries to add top concept of next ontology. Next, the algorithm tries to add all the child concepts of current concept by recursively calling itself. Since subClassOf relation is used here, efficient pruning can be employed for the rule that does not satisfy size constraint as well as all the rules constructed using its child concepts.

2.3.4 Rule Selection

The ruleset generated by rule construction algorithm is extremely big and may contain uninteresting, overlapping rule. These rules need to be filtered out. Here, in rule selection algorithm all these unwanted rules are filtered out using WRAcc (weighted relative accuracy) heuristic [10]. Rule selection algorithm is shown in Figure 1(b). The formula for calculating WRAcc is defined as:

$$WRAcc(C \leftarrow \ Cnd) = \frac{n(Cnd)}{N}.\left(\frac{n(Cnd\Lambda C)}{n(Cnd)} - \frac{n(C)}{N}\right)$$

Where N is number of all examples, n(C) is the number of examples of class C, n(Cnd) is the number of all covered example and n(CndΛC) is the number of all positive examples of class C.

```
function construct(rule, conj, k):
 # rule - the rule to specialize.
 # conj - the concept to add to the rule.
 # k - 'conj' is from the k-th ontology.

 # The set described by the current rule.
 newSet = intersect(set(rule), set(conj))

 # Is the set big enough?
 if newSet.size > MIN_SIZE:
   rule.add(conj)
   if clean(rule).size < MAX_TERMS and
      clean(rule).size > 0:
     rules.add(rule)

   # Can the rule be extended?
   if rule.size < max(MAX_TERMS, MAX_ONT):
     construct(rule, ontologies[k+1], k+1)
     rule.remove(conj)

     # Extend the rule with all successors.
     for each child in children(conj):
       if set(child).size > MIN_SIZE:
         construct(rule, child, k)

 return rules
```

(a) Rule Construction procedure of SDM-SEGS[4]

```
function ruleSelection(examples, k):
 # examples - example set.
 # k - an example can be covered max k times.

 # Construct the rule set.
 ruleSet = construct([], ontologies[0], 0)
 resultSet = []
 repeat
   # Currently best rule according to WRAcc.
   rule = bestRule(ruleSet)
   resultSet.add(rule)
   # Decrease weights of covered examples
   # and remove examples covered k times.
   decreaseWeights(examples, rule, k)
 until examples == [] or ruleSet == []

 # Re-compute the WRAcc, ignore the weights.
 for each rule in resultSet:
   rule.score = WRAcc(rule)

 return resultSet
```

(b) Rule Selection procedure of SDM-SEGS[4]

Figure 1: SDM-SEGS procedures

Rule selection proceeds as follow: (1) set of all examples are assigned with a counter k. (2) All possible rules generated by rule selection is sorted in descending order on the basis of WRAcc value. (3) Each time top rule from the sorted list is selected and counter of all covered example is decremented until counter value is zero. (4) As the counter value of a particular example becomes zero, it will be removed from the example set.

This procedure continues until example set is empty or all possible rules are enumerated.

3 Implementation

SDM-SEGS system (Figure 2) accepts dataset in form of dataframes, ontologies in form of RDD and example-to-ontology mapping as inputs. System consists of mainly two components, rule construction and rule selection. Rule construction generates all possible rules of given subgroup based on received inputs. The rules are generated with following constraints: all true examples satisfying the rule should be greater than minimum threshold value and the number of conjunctive terms in the rule should be less than a given maximum value. Rule Selection receives all possible rules generated by rule construction and generates accurate rule set based on WRAcc value of each rule.

SDM-SEGS is implemented in Scala using Apache Spark framework. Apache Spark[11] is a fast and general engine for large-scale data processing. It run programs up to 100x faster than Hadoop, supports multiple programming languages (Scala, Java, Python and R), combines

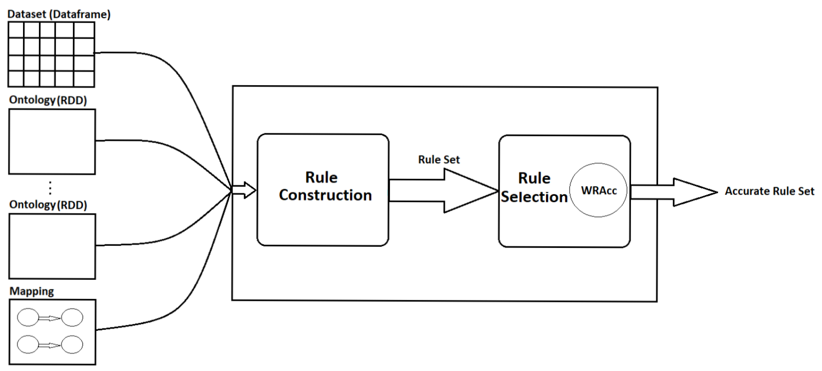

Figure 2: Workflow of SDM-SEGS

SQL, streaming, and complex analytics[11]. Scala programming language is preferred because it offers concise syntax, easy to learn and compatible with Apache Spark. The code achieves scalability since input dataset (dataframe) and ontologies (RDD) are distributed across multiple worker nodes in a cluster. Since, ontologies are very small files compared to dataset, they are set as broadcast variables so that they are readily available in all worker nodes. This enables operation on ontologies efficient and achieves better performance.

4 Evaluation

All the experiments are performed on a single system with configuration of Intel Core i5 (8th gen) processor and 8 GB RAM. We used a sample dataset of big spenders with 15 positive class examples and 15 negative class examples along with the ontologies (geography, occupation and banking service). Experiments were repeated 5 times on the same two sets (splits) of positive/negative examples. For each split top 10 rules were produced and all the rules were validated and verified.

Using [10] we have used following measures for evaluation:

1. The *average rule coverage* (COV), it calculates the average of ratio of covered examples and total number of examples n(Cnd)/N over a given rule set.

2. The *overall support* (SUP), it is the ratio of positive examples satisfied by the rule and total number of positive examples.

3. The *average interestingness* (WRACC), it is the average WRACC over a given rule set.

Table 1 represents achieved results produced by the algorithm for average COV, average SUP and average WRACC.

It took approximately 6-7 minutes to generate all rules and to produce top 10 rules, using the system with above mentioned configurations. Since, the program is written using distributed data structures like RDDs, DataFrames provided by Apache Spark framework, we would be able to process large datasets in distributed manner. More efficiency and scalability can be achieved by running the program on a cluster.

Split	Average COV	Average SUP	Average WRACC
1 (15 +ve & 15 -ve examples)	0.523	0.570	0.024
2 (14 +ve & 14 -ve examples)	0.525	0.577	0.026
3 (13 +ve & 13 -ve examples)	0.633	0.654	0.011
4 (12 +ve & 12 -ve examples)	0.689	0.708	0.009
5 (11 +ve & 11 -ve examples)	0.848	0.879	0.015

Table 1: Average COV, average SUP and average WRACC of SDM-SEGS

5 Project Timeline

Week	Tasks	Responsible person
Week 1 (22/11/2017 - 29/11/2017)	Understanding of assigned paper and related papers	Livin Pardeep
Week 2 (29/11/2017 - 05/12/2017)	Project planning	Livin Pardeep
Week 3 (05/12/2017 - 12/12/2017)	Dataset acquisition and pre-processing	Livin Pardeep
Week 4, 5, 6 (12/12/2017 - 02/01/2018)	Implementation of rule construction	Livin
Week 4, 5, 6 (12/12/2017 - 02/01/2018)	Implementation of rule selection	Pardeep
Week 7 (02/01/2018 - 09/01/2018)	Integration of rule construction and rule selection	Livin Pardeep
Week 8 (09/01/2018 - 16/01/2018)	Optimization of code	Livin Pardeep
Week 9 (16/01/2018 - 23/01/2018)	Changes in code w.r.t feedback from meeting	Livin Pardeep
Week 10, 11 (23/01/2018 - 08/02/2018)	First draft of lab report & Preparation for exams	Livin Pardeep
Week 12 (08/01/2018 - 15/02/2018)	Evaluation and testing	Livin Pardeep
Week 13 (15/01/2018 - 20/02/2018)	Final changes in lab report	Livin Pardeep

Table 2: Project timeline

References

[1] Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P. (1996) From data mining to knowledge discovery in databases. *AI Mag., 17,* 37-54.

[2] Klsgen, W. (1996) *Advances in Knowledge Discovery and Data Mining.* American Association for Artificial Intelligence, MenloPark, CA, USA.

[3] Wrobel, S. (1997) An Algorithm for Multi-relational Discovery of Subgroups. *Proc. 1st European Conf. Principles of Data Mining and Knowledge Discovery (PKDD97),* Trondheim, Norway, June 24-27, pp. 78-87. Springer, Berlin, Germany.

[4] A. Vavpetic and N. Lavrac (2012) Semantic Subgroup Discovery Systems and Workflows in the SDM-Toolkit. *The Computer Journal, 2012.* 56(3):304–320.

[5] Lavrac, N., Vavpetic, A., Soldatova, L., Trajkovski, I. and Kralj Novak, P. (2011) Using Ontologies in Semantic Data Mining with SEGS and g-SEGS. *Proc. Int. Conf. Discovery Science (DS11),* Espoo, Finland, October 5-7, pp. 165-178. Springer, Berlin/Heidelberg, Germany.

[6] Lavrac, N., Novak, P., Mozetic, I., Podpecan, V., Motaln, H., Petek, M. and Gruden, K. (2009) Semantic Subgroup Discovery: Using Ontologies in Microarray Data Analysis. *Proc. Annual Int. Conf. IEEE, Engineering in Medicine and Biology Society (EMBC09),* Minneapolis, USA, September 2-6, pp. 5613-5616. Institute of Electrical and Electronics Engineers, NewYork, USA.

[7] Liu, H. (2010) Towards Semantic Data Mining. *Doctoral Consortium of the 9th Int. Semantic Web Conf. (ISWC10),* Shanghai, China, November 711. `http://data.semanticweb.org/conference/iswc/2010/paper/448`.

[8] Subramanian, A. et al. (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc. Natl Acad. Sci. USA, 102,* 15545-15550.

[9] Kim, S.Y. and Volsky, D. (2005) PAGE: parametric analysis of gene set enrichment. *BMC Bioinf., 6,* 144-155.

[10] Lavrac, N., Kavek, B., Flach, P. and Todorovski, L. (2004) Subgroup discovery with CN2-SD. *J. Mach. Learn. Res., 5,* 153-188.

[11] Apache Spark Apache Spark is a fast and general engine for large-scale data processing. `https://spark.apache.org`.