Learning Functionality Graph of Products: Substitute and Complementary Relations on Amazon

Amit Sharma*
Department of Computer Science
Cornell University
asharma@cs.cornell.edu

December 18, 2010

Abstract

In general, machine learning algorithms focus on predicting outputs from 'flat' data. However, increasingly a lot of the data available inherently has a relational structure, either explicit or implicit. This implies that the data values are not independent from each other. Further, individual data values may be related to each other through different classes of relationships. These relationships,in turn, may not be independent of each other and may influence the existence of other relationships between the same or other pairs of nodes. We consider the inherent challenges that such a problem formulation presents and try to use a relational Markov network model that learns over the individual attributes as well as the relations among the entities. We apply the model to a dataset containing Amazon product items, which has complementary and substitute item relations. A major focus on efficacy of the approach is on the consistency of the resultant functionality relationship graph, in addition to accuracy in relation predictions. Through carefully designed evaluation scenarios, we demonstrate the superior performance of the RMN model as opposed to 'flat' classifiers.

1 Introduction

Consider the variety of products on sale at the online shopping website, Amazon. Every product has an intrinsic functional value. Given a desired functionality requirement, it is generally possible to satisfy it using an appropriate combination of individual products. However, with a varied set of products, finding

*This report is submitted as a part of the CS6780 Machine Learning course at Cornell University.

out such an optimum combination is non-trivial, since we need to ascertain the smallest(or the cheapest) set such that none of the products conflict with each other. It is clear that to solve the problem, we need to model the functionality of every product in an computable way, in addition to satisfying the constraints imposed by the selection of each constituent product.

Since the functionality of a product an abstract concept, there can be different interpretations to it.One could approximate the functionality of a product by the category it belongs to. This approximation only works well when there are a small number of coherent products in a category. More often, categories include a vast collection of products sharing some common features but different in many other aspects of features and usability. Another way to learn functionality would be by processing the accompanying product information text and forming a model about the functionality for each product. A lot of information can also be inferred by analyzing the functionality relations between products. These relations can be divided into two main classes-substitute and complementary relations. Intuitively, these relations capture relative functionality among any two products. We focus our attention on the functional relations.

For every product, there would be certain other products which serve exactly the same functionality, and hence can be used as substitutes. Similarly, there would be other products which would add to the value of the product, and hence act in a complementary fashion. Note that the linkages are defined at two hierarchical levels, category and product. All products of category 'mouse' are expected to be complementary with all products of category 'Laptop'(category level). However, a mouse made specifically for an Apple Macbook might not be complementary to all laptops(product level). Our goal in this project is to predict the latter, the fine-grained relationships between individual products.

More specifically, given a set of items, our task is to construct the corresponding functionality graph, consisting of Substitute and Complementary links.This involves predicting a composite structure of consistent links, and presents unique challenges. The first challenge is the relational nature of the dataset, which implies that the individual relations are not independent of each other. The second is the semantic constraints that the nature of the domain force on to the graph. For instance, we cannot have a cycle of Complementary relations, and we need to somehow model these constraints into the learning model.

We choose a Relational Markov Network(RMN) to model the relationships. By a careful selection of clique templates, we are able to ensure both dependence among the relation types as well as the constraints regarding sanity of the product tree. In the next sections, we define the problem statement and explore some preliminaries, before discussing RMNs.

2 Problem Definition

The problem is best illustrated with the help of an example. Consider a small sub graph of the products dataset as shown in Fig. 1.

The links show the relationships between a set of products. The two major

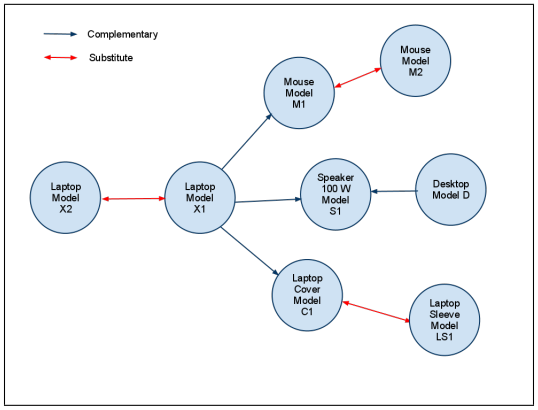

Figure 1: A products subgraph showing the relations

relationships are Substitute and Complementary. For instance, a laptop may be a substitute for another model of a laptop, but a mouse or a speaker will be complementary to these items. Existence of a link between two products implies a certain property on the products, and gives us a better idea of the pair's relation with other products in the graph. Note that both of these relationships have some inherent properties which are not necessarily similar. For example, the Substitute relation is transitive and symmetric, but the same is not necessarily true for the Complement relation.
Consider a set $V = \{v_1, v_2, v_3 ... v_n\}$ of n products. For each product, we have a set of p attributes, which are denoted by the set $A_i = \{a_{i1}, a_{i2}, ... a_{ip}\}$. In addition, each product is related to other products by means of Substitute and Complement relations, which we define by S and C as below.
$s_{ij} = \{(v_i, v_j) : v_i \text{ and } v_j \text{ are substitutes}\}$
$c_{ij} = \{(v_i, v_j) : v_i \text{ is complementary to } v_j\}$

This gives us a graph $G(V, E)$ where $E = S \bigcup C$. Note that this is not a graph in the strict sense, since C edges are directed, while the S edges are undirected. The problem of predicting the new links can now be stated as follows.
DEFINITION 1: Given a products graph $G(V, E)$ and a separate set of products P disjoint with V, predict the S and C relations in the new graph $G' = (P, E')$.

If we consider the edges as data points for learning(and ignore the relational dependencies), then the latter formulation is a standard classification problem with three output classes = {Substitute, Complementary, None}.

2.1 Related Work

Currently, the taxonomies used to classify products achieve it on a very coarse basis(not per product basis), using either the subcategories of products , or

their manufacturer or any other tangible attribute. Amazon and other merchants have systems that try to suggest the functional relations, but most of the methods are based on the statistics of users' action, rather than modelling of the inherent functionality.

At a more conceptual level, the goal here is to predict a structured output based on relational data. Recently, a lot of interest has been shown in the machine learning community over relational learning. Owing to the relational nature of the data, many relational learning models extend graphical models to model the relations between data points [6, 3, 5]. However, none of the previous work on relational learning have had their focus on predicting consistent structural output, which is our goal in this paper.

2.2 Contribution

A new dataset for product domain was collected. A previously existing model,Relational Markov Network, was applied to the novel products domain to predict the functionality graph. It was ensured that the final graph is consistent. To the best of our knowledge, our work is the first attempt at learning functional relationship among products using relational learning.

3 Preliminaries

3.1 Data Collection

A major obstacle was the absence of relevant data about the relationships that we plan to capture. Hence, the first task was to collect data about the relationships between products. Amazon seemed to be a good choice, because they publicly show recommendations to users (based on the product being viewed) on their website. Along with basic attributes of each product, we collected items marked as 'Frequently Bought Together' and 'What did users ultimately buy?'.

Note that the data collected does not give an accurate picture of the substitute and complementary relationship. Amazon uses the statistics related to its customer' activities to present these suggestions, and so this information is inherently noisy. We will discuss later how our model formulation handles the noise in data.

We concentrated on the Electronics store of Amazon. The following information about a product was collected, using a combination of web scrapping of the Amazon website plus Amazon's Product API calls.

Name, ASIN code (Unique Amazon ID), Price, Manufacturer, Category, Hardware Platform, Operating System, Department, Product SubCategory, 'Tags associated with the product', 'Frequently Bought Together items'(Complementary Products), 'What did Customers ultimately buy after viewing this item' (Substitute Products), 'Ancestor categories of the product'

Though the text description of the product may allow us a more accurate (and less noisy) model of the utility of the product, we decided not to include it

Table 1: Characteristics of data

Property	Nodes	Edges
PRODUCT TYPE		
– Computer	2505	
– Consumer Electronics	1007	
– Software	911	
– Others	5577	
PLATFORM		
– PC	6281	
– Mac	1207	
– Others	2512	
CLASS		
– Substitute		35540
– Complementary		33446
TOTAL	10000	68986

mainly because of the variation in the description text size and quality among products of different manufacturers. Note that 'Hardware Platform' and 'Operating System' are domain-specific features.

3.2 Relational Domain

We use the analogy of relational database systems to specify the problem domain. Our schema consists of a single entity type, Product-Pair(**P**). Each product-pair P is associated with three sets of attributes, content attributes P.X(e.g. categories of constituent products), relation attributes P.R(e.g. link to other product-pairs sharing a product) and label attributes P.Y(i.e. Substitute, Complementary, None). An instantiation I of the schema specifies the set of actual entities and the values of attributes for each entity. We will use $I.X$, $I.Y$, and $I.R$ to denote the content, label, and reference attributes in the instantiation I; and $I.x$, $I.y$, and $I.r$ to denote the values of those attributes.

4 Modelling the Problem

One of the major drawbacks of standard machine learning algorithms that we tried was that they are not able to model the topographic features. So we now try to focus on the problem using a learning model that can learn over relations present between entities. We choose the Relational Markov Network [3] for reasons underlined in the Appendix. In particular, the formulation involving

cliques lends itself well to the concept of patterns in relationships, which are useful for the products network.

An RMN requires the specification of domain class and the relations that work on instances of that class. For the product relationship problem, we define the domain as the set of all possible edges between products. Note that this includes actual as well candidate edges, which we would like to predict. We specify the relationship in the form of clique templates, which translate to cliques in the unrolled Markov network. More precisely, a relational clique template specifies a set of cliques in the Markov network. Each clique template defines the rules for a clique to be included in the template. Intuitively, a clique template can be thought of as an assertion about the classes of the involved nodes, given some conditions pertaining to the nodes hold.

DEFINITION: A relational Markov network $M(\mathbf{C}, \Phi)$ specifies a set of clique templates $\mathbf{C}$ and corresponding potentials $\Phi = \{\phi_c\}_{C \in \mathbf{C}}$ to define a conditional distribution:

$$P(I.y|I.x, I.r) = \frac{1}{Z(I.x, I.r)} \prod_{C \in \mathbf{C}} \prod_{c \in C(I)} \phi(I.x_c, I.y_c)$$

where $Z(I.x, I.r)$ is the normalizing partition function (all other variables are as defined before in Section 3.2):

$$Z(I.x, I.r) = \sum_{I.y'} \prod_{C \in \mathbf{C}} \prod_{c \in C(I)} \phi(I.x_c, I.y'_c)$$

4.1 Clique Formulation

We considered the important properties of the Substitute and Complementary relations to arrive at the cliques templates to use. In choosing the templates, there is a direct trade-off between expressiveness of the templates and tractability of the Markov network generated. Substitute relation follows commutative and transitive, and reflexive properties. For complementary relations, we note that the probability of complementary relations increase between a pair of products if there exists a similar edge to one of them from the other's substitute product.

We choose a size of maximum 3 for the templates, since that is the minimum required to correctly encode the transitivity relation. After a careful analysis of the products network, we propose the use of the following two clique templates. These templates cover most of the dependencies enumerated above, plus have the advantage of being compact in their representation.

We define the cliques as follows.

1. For every 3 nodes which are good candidates of substitution amongst each other, define a clique for the edges connecting them.

2. For every 3 nodes such that exactly two of them are good candidates for substitution, and the other node is a good candidate as a complement to

either of them, define a clique connecting the three possible edges of the triangle.

We determine a 'good candidate' by defining a score function over all the attributes of each product-pair for Substitution and Complementary links separately. Here, the first template encodes the substitute relation properties. The second one is used for the prediction of complementary relations based on existing relations of substitutes of the same product. Though the formulation seems to suggest $O(n^3)$ cliques for the network, the formulation of properties for good substitution candidates allows for a more computable number of cliques.

4.2 Learning and Inference

We maximize the likelihood of the conditional probability to return the optimal values for weights(w) of the clique potentials. To help avoid overfitting, we assume a prior over the weights and use maximum a posteriori(MAP) estimation [3]. Writing the potential function as $\phi(v_c) = \exp(w.f(v_c))$, we write the likelihood of the data:

$$L(w, d) = w.f(I.y, I.x, I.r) - \log Z(I.x, I.r) - \frac{||w||_2^2}{2\sigma^2} + C$$

Note that here d is the only data point, which is the instantiation of the whole product network.The gradient is given as:

$$f(I.y, I.x, I.r) - E_w[f(I.y, I.x, I.r)] - \frac{w}{\sigma^2}$$

where the expectation $E_w[f(I.y, I.x, I.r)]$ is :

$$\sum_{I.y'} f(I.y', I.x, I.r) P_w(I.y'|I.x, I.r)$$

Here, the sum over $I.y'$ involves an exponential number of assignments to all the label attributes in the instantiation. Hence, we need to compute the expectation over the joint assignments to all the entities together. For a large network such as in the products domain, exact inference would turn out to be intractable. Hence we use an approximate method, Belief Propagation(BP) for inference on the unrolled Markov network.

4.3 Dealing with Noisy Data

We now show how specifying the model using the clique formulation allows us to effectively deal with noisy data (as long as it is not very frequent). Suppose in the training data, an Windows mouse was associated with an Apple laptop as complementary product. Although from an ontological point of view, the relation seems correct, it is not valid for the specific pairs of items, because they are incompatible. Since the clique structure is expected to have a condition stipulating that the 'Hardware Platform' attribute for complementary links

should be the same, this relation edge will not be included in any of the cliques. This implies that the presence of this spurious data point will not affect the computation of the probabilities of other relations in the Markov network.

5 Graph Consistency

The nature of the problem demands that the relationships generated must always follow a tree structure for complementary links. Cycles are allowed for substitute links, but only if all the links in the cycle are substitution relations. Our current formulation of RMN does not enforce any of these checks explicitly. However, there are soft constraints in both clique templates that decrease the probability of certain inconsistent configurations(such as a cycle of three Complementary relations). Surprisingly, testing on the data has shown that the conditions are obeyed for most parts of the graph generation. Intuitively, it can be imagined that the general structure imposed by the cliques propagates these properties to the whole graph.Ensuring graph consistency is then a simple process of running a consistency check on the output graph, and classifying the least probability edges among the conflicting edges as $None$.

For the purpose of comparison, we also encoded the structure constraints explicitly in the network. A clique template was included in the model for all the cases that cannot occur in a consistent functionality graph. These cliques were assigned low potentials(≈ 0) to prevent their occurrence. We call this model, RMN with Explicit Checking(EC), and will compare the results in the next section.

6 Evaluation

A variety of scenarios were used to evaluate the Relational Model. Most of these concern varying the amount and structure of train data available. Given the related nature of the data points, the model was tested against a wide setting of train/test data selection.

- **Incremental Addition:** Given a product graph complete with nodes and relations, predict the edges of a new node to be added.
- **Missing Links**: Randomly select k edges and k non-edge node-pairs and remove them from the graph. Train the model and predict the classes of these node-pairs.
- **Empty Graph:** Given a set of nodes, predict the relationship graph between them.
- **Sparse Graph:** Given a set of nodes and a few relationships, predict the whole relationship graph.

Table 2: Results of Preliminary Algorithms

ALGORITHM	Substitute Accuracy	Complementary Accuracy
Multinomial Logistic	94.5	84.3
K-nearest Neighbors	85.8	71.0
Bayes	82.4	76.4
Decision Tree	91.6	63.3

6.1 Preliminary Algorithms

Out of the above, Missing Links scenario lends itself well to standard machine learning formulation. If we consider the edges as data points for learning, and the type of relation as the output class, then this formulation is a classification problem with three output classes = {Substitute, Complementary, None}. Hence, we ran a baseline set of algorithms against the data for this scenario- Multinomial Logistic Regression, Bayes Classifier, K-nearest Neighbors and Decision Trees. For all these algorithms, the data was suitably parsed and appropriate features were assigned. Note that since none of these models capture the underlying product graph, no topological features could be used. Inherently, each of these models treat each edge as an independent data point, and tries to predict its class. We present the results of these algorithms in Table 2.

It is important to note here that the difficulty of the problem varies with each scenario, increasing in the order: Incremental Addition, Missing Links, Sparse Graph and finally Empty Graph. The key difficulty posed in each of the problems is the prediction of consistent edge relations. As the number of potential edges in these scenarios increase, the number of possibly valid and consistent edge configurations increases exponentially. In fact, Empty Graph is the hardest since it requires to predict the relationships among nodes without any prior evidence in the graph.

6.2 RMN evaluation

First of all, RMN model was tested against the baselines algorithms for the Missing Links scenario. The data was divided into 5 parts, and each part was individually divided into test and train data. For the Incremental Addition scenario, we divided the whole dataset into two parts(80-20), and trained the model using the majority of the data. Performance was averaged over all the individual runs for each test node. The results are presented in Figure 6.2.

The empty graph scenario demanded that we have two completely independent graphs. For this, we randomly divided the products into two sets, and removed all the edges between the two sets. Also, for one set, we removed all the edges between its nodes too. Learning was performed on the first set, and

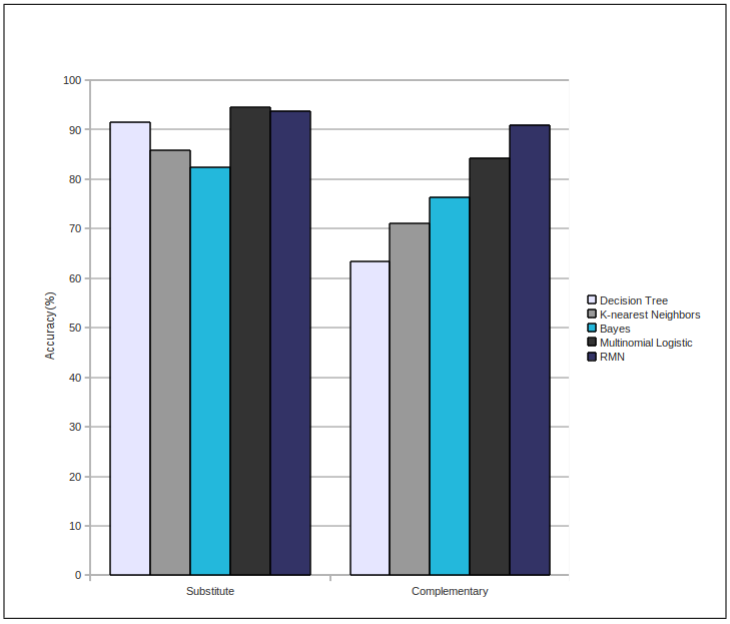

Figure 2: Performance of various algorithms on the Missing Links scenario

inference on the other. The sparse graph scenario was similar to the Empty Graph case, except that a few of the edges in the test set were retained. We fixed this number to 10 percent of the number of nodes in the test data. The results are presented in Table 3.

6.3 Discussion

In the Missing Links scenario, RMN outperforms the other baseline models as expected. In particular, multinomial regression performs equally well for the Substitute prediction, but has a far lower accuracy for Complementary links. This can be attributed to the fact that Substitution relation is just a measure of nearness of the product attribute vectors in the attribute space, and hence can be accurately predicted by 'flat' classifiers. The real improvement of the model is in predicting the Complementary links. Similar results are obtained for the Incremental Addition scenario.

The Empty Graph(EG) and Sparse Graph(SG) scenarios present interesting results.The accuracy drops for Complementary prediction in the EG scenario. The RMN is a highly relational model, and the edges chosen by the model depend a lot on the already present edges. It appears that EG scenario offers a wide scope for RMN to make errors in the beginning, which then translates to a lower accuracy in classification overall. The SG scenario (with $0.1N$ edges) however, performs with an accuracy comparable to the Missing Links case.This

Table 3: RMN Performance: Different Scenarios

SCENARIO	Substitute Accuracy	Complementary Accuracy
Incremental Addition	95.5	91.3
Missing Links	93.8	91.0
Empty Graph	91.4	79.4
Sparse Graph	92.3	87.3
Empty Graph(with EC)	90.9	80.3
Sparse Graph(with EC)	92.9	87.1

is an important observation, and leads us to conclude that the RMN requires only a few edges to be initialized in a new domain to be more effective.

The EG and SG scenarios with Explicit Consistency(EC) have roughly the same performance as standard EG and SG on the dataset. Further, since we enumerate all the possibly inconsistent cliques, the running time becomes intractable for large networks. We conclude that the standard RMN learns the constraints implicitly well and for most cases, explicit consistency is an overkill and can be avoided.

7 Conclusions and Future Work

In this paper, we have presented an application of Relational Markov Model to the products domain. Experiments indicate that prediction of the functionality graph is highly accurate using RMN, except in the Empty Graph case. However, it must be emphasized here that most practical applications of the functionality graph concern predicting the relations of a new unseen product, or suggesting possible relations for a group of products. In all such scenarios, our experiments show RMN to be highly effective.

Future work includes testing variations of the RMN model for efficiency, specifically clique potentials and features. To possibly improve the results of the Empty Graph scenario, we would want to try online learning and analyze the performance. For the Sparse Graph scenario, we would also like to investigate the effect of number of initial edges labelled on the performance of the relational model, and try to find an optimal ratio.

Appendix:Relational Learning-A Brief Survey

Because of the inherent dependence between the various data points, it appears that a model with knowledge of the relations might perform better than the standard learning algorithms. In recent years, a lot of interest has been

shown in learning models over relational data, encompassing the field Statistical Relational Learning(SRL). Three major models under this framework are the Markov Logic Networks, Probabilistic Relational Models and Relational Markov Networks. A major part in this project was to ascertain which of these models would give the best results for the products domain.

Probabilistic Relational Model

A PRM [5] specifies a probability distribution over a particular instance of the graph by specifying a Bayesian network-like template-level probabilistic model for each entity type. It can be thought of as a relational extension to Bayesian networks.Given a particular instantiation graph, the PRM induces a large Bayesian network over that instantiation that specifies a joint probability distribution over all attributes of all of the entries. However, this formulation runs into a problem if the input graph has a possibility of having cycles. In the products domain, there could be cycles between items that are substitutes of each other, which would in turn lead to cycles in the induced Bayesian network. This would not lead to a coherent probabilistic model. Even if we treat the relations themselves as random variables [4], we need to consider a big performance slowdown due to N^2 number of links between any N nodes.

Markov Logic Network

Markov Logic networks [6] differentiate the inherent constraints in the problem and the probability maximization. It is a combination of first-order logic and probabilistic graphical model in a single representation. It can be viewed as a generalization of first order logic, in which every clause now has a finite probability of being true. Although this model allows for a very expressive framework, it is also a very complicated model to build since we need to enumerate all the clauses. Especially for the products domain, this model does not suit well to encode the exact nature of the properties of relations and constraints into first order logic.

Relational Markov Network

This model tries to combine the relational concepts of databases to the probabilistic graphical model formulation. It specifies a set of clique templates based on the relations among entities, which then corresponds to a set of cliques in the unrolled Markov network. This model overcomes the disadvantage of PRMs in that it is an undirected model and can be used to model cycles in the graph. Further, cliques allows for a succinct representation of the correlation between the attributes of entities and the relations in which they participate.

References

[1] Structural Logistic Regression for Link Analysis, Alexandrin Popescul, Lyle H. Ungar. *2nd Workshop on Multi-Relational Data Mining (MRDM 2003)*

[2] Link Prediction using Supervised Learning Al Hasan, Chaoji et al. *Proc. of SDM 06 workshop on Link Analysis, Counterterrorism and Security* , 2006

[3] Link Prediction in Relational Data *Proceedings of Neural Information Processing*, 2003

[4] L. Getoor, E. Segal, B. Taskar, and D. Koller. Probabilistic models of text and link structure for hypertext classi?cation. In Proceedings of the IJCAI01 Workshop on Text Learning: Beyond Supervision, 2001.

[5] N. Friedman, L. Getoor, D. Koller, and A. Pfe?er. Learning probabilistic relational models. In Proceedings of the International Joint Conference on Artificial Intelligence, 1999

[6] Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. Mach. Learn. 62, 1-2 (February 2006), 107-136. DOI=10.1007/s10994-006-5833-1 http://dx.doi.org/10.1007/s10994-006-5833-1