TOWARDS ONLINE COMPOSITION OF PMML PREDICTION MODELS

Diana Gorea
Faculty of Computer Science, University "Al. I. Cuza", 16 G-ral Berthelot, 700483 Iasi, Romania

Keywords: Prediction model composition, PMML, online scoring, web services, XML.

Abstract: The paper presents a general context in which composition of prediction models can be achieved within the boundaries of an online scoring system called DeVisa. The system provides its functionality via web services and stores the prediction models represented in PMML in a native XML database. A language called PMQL is defined, whose purpose is to process the PMML models and to express consumers' goals and the answers to the goals. The composition of prediction models can occur either implicitly within the process of online scoring, or explicitly, in which the consumer builds or trains a new model based on the existing ones in the DeVisa repository. The main scenarios that involve composition are adapted to the types of composition allowed in the PMML specification, i.e sequencing and selection.

1 INTRODUCTION

In general the composition of prediction models can be realized in various ways. They all have in common the goal of making predictions more reliable. Composing several prediction models means merging the various outputs into a single prediction. Several machine learning techniques do this by learning an ensemble of models and using them in combination. The most popular are bagging, boosting and stacking (Ian H. Witten, 2005).

Bagging applies an unweighted voting scheme on the outcomes of different classifiers built on possibly different data sets (which can be obtained by resampling the original data set). In the case of numeric prediction, instead of voting on the outcome, the individual predictions, being real numbers, are averaged. The component models are usually built separately.

Boosting also uses voting or averaging schemes to combine the outcomes, but, unlike bagging, it uses weighting to give more influence to the more successful models. Furthermore the process is iterative, each component model is built upon the previous model and therefore influenced by their performance.

Stacking applies on heterogeneous classifiers and trains a new meta learner on the predictions of the component classifiers using a validation data set. The meta learner can be of various types depending on the set of attributes used for meta learning. Some meta-learners use only the class predictions of the component models for training, while others use both the class predictions and all the original input attributes. It applies both to categorical and numeric predictions.

Another approach which is mostly useful in distributed data mining is combining models with various levels of granularity. For instance, it might be the case that a model classifies an instance at a coarse level, while another model does it with a finer granularity. One can use the first model to tag the instances with a more general class and then to use a specialized classifier for each of the resulted groups. This technique is also useful in classification when an algorithm cannot predict multi-class attributes, such as standard SVM. Another direct use of this technique is model selection in PMML, which is described in 2.

In the distributed data mining model different models might be built on vertically fragmented data (usually that reside at different sites). Each of the individual models is built on projections on the same relation, but unable to detect cross-site correlations. A meta-learning approach has been proposed in (Prodromidis et al., 2000) that uses classifiers trained at different sites to develop a global classifier.

Another possibility to combine data mining models is to adjust a given model to a consumer's specific needs - *model customization*. A model consumer repeatedly uses the model in its knowledge integration processes and may collect information on its perfor-

Gorea D. (2008).
TOWARDS ONLINE COMPOSITION OF PMML PREDICTION MODELS.
In *Proceedings of the Third International Conference on Software and Data Technologies - ISDM/ABF*, pages 300-303
DOI: 10.5220/0001890503000303

mance. A new model can be constructed based on the specific needs for the consumer. The same applies to refreshing a model to reflect the new trends in the data. This is related to the concept of *incremental data mining*, which was introduced in (M.Harries et al., 1998). A related approach can be found in (Kuncheva, 2004), where strategies for building ensembles of classifiers in non-stationary environments in which even the classification task may change are presented.

The current work aims to provide a general context in which composition of data mining models can be achieved within the boundaries of an online scoring system called DeVisa. While in (Gorea, 2008) and (DeVisa, 2007) a general description of the system is given, the current paper focuses on the model composition aspect. DeVisa only stores the prediction models expressed in PMML (PMML, 2007), not the original training data. Therefore we consider the training data as not being available. Subsequently the model composition in DeVisa is limited to certain techniques which are described in 2. However the consumer application can build new models based on the existing DeVisa models and its own validation set.

2 DEVISA COMPOSITION OF PREDICTIVE MODELS

DeVisa supports two types of composition methods described in the PMML specifications: *model sequencing* and *model selection*. In its current version (3.2), PMML supports the combination of decision trees or rules and simple regression models.

Model sequencing is the case in which two or more models are combined into a sequence where the results of one model are used as input in another model. Model sequencing is very often an intrinsic part of a model, namely a transformation function. For instance, a supervised discretization algorithm is applied to a certain attribute, which is described within a transformation dictionary, or missing values for an attribute are filled using a transformation function made of decision rules. Model selection is when one of many models can be selected based on decision rules. A common model selection method for optimizing prediction models is the combination of segmentation and regression.

Although the producer applications can upload composite models in DeVisa, in this section we focus moreover on the situations in which DeVisa is responsible for composing the models. Depending on the moment when the composition process occurs, we can further classify the composition in DeVisa in *implicit* or *explicit* composition.

Implicit composition is the situation when the models are composed within the orchestration of the scoring or search service (see 3).

A *scoring goal* (query) is a tuple $(MSpec, R)$, where

1. *MSpec* is the *model specification*, defined as

$$MSpec ::= \{MRef\} \mid (\{Filter\}, SRef \mid S[, DRef])$$

The model specification has several instances:

Exact model case, in which exact references to one or more DeVisa model that the consumer wishes to score on is given via $MRef$.

Exact schema case, in which the consumer gives an exact reference to a mining schema $SRef$ and wishes to score on the models complying to that schema. However, an additional set of filters corresponding to the properties that the model needs to conform to can be specified via the $Filter$ element.

Match schema case, in which S describes a mining schema that needs to be matched against one or more in the DeVisa Catalog. To restrict the search, an existing DeVisa data dictionary can be optionally referenced. A reference to an ontology in order to explain the terminology can be included. Also an optional $Filter$ element can be specified.

2. R is the dataset to score.

The implicit model composition is applicable in two situations, given that the consumer allows scoring on composite models:

1. More models complying to *MSpec* can be found;
2. No model complying to *MSpec* can be found.

The first situation can occur in all the model specification instances (exact model, exact schema or match schema).

In the first two instances, given the models or schema reference, an existing data dictionary is implicit. In the match schema case a mining schema S and a reference to a data dictionary in DeVisa, $\mathcal{D}$ should be provided.

In the exact model case, all the referred models are retrieved. In the exact schema case the engine finds all the models complying to the specified schema. In the match schema case, the engine tries to find one or more models that match S. The composition described below applies to the situation in which more models satisfy the requirements. They are combined to give the best prediction as follows. The composer component of the engine (see 3.1) scores on all the models and then applies a voting procedure (similar to the bagging approach) and returns either the outcome that has the highest vote (in the case of categorical predictions), or the average (in the case of numeric

predictions). Note that if the model composition is not allowed the engine builds a query plan and executes it against the retrieved model (or models) in the repository - classical scoring scenario.

In the match schema case, the names of the attributes in S should either be among the attribute names in $\mathcal{D}$, or they should refer to the same terms in an ontology/taxonomy, so that a clear mapping can be made. Thus we refer to the case when no model applicable on S can be found. Then the engine is going to invoke the composer module that attempts to build a new model from the models complying to the data dictionary $\mathcal{D}$ via composition. An example can be seen in Figure 1, which depicts a sequencing composition. The two DeVisa classification models ϕ_1, ϕ_2 are defined on the schemas $S_1(\{A,B,D,F\},P_1), S_2(\{E,F,G\},P_2)$ in the data dictionary $\mathcal{D}$. The scoring goal specifies a mining schema $S = (\{A,B,C,E,G\},P)$. The composer attempts to build a new model by sequencing ϕ_1 and ϕ_2.

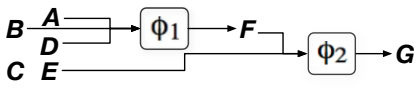

Figure 1: An example of implicit sequencing of models.

In the implicit composition scenario the PMQL engine builds the sequenced model in order to score on the dataset provided in the scoring request. It then stores the model back in the repository for future use. **Explicit Composition** is when a DM consumer explicitly specifies in its goal that a composition is desired. The composition query usually includes the models to be composed, the composition method and a validation data set. DeVisa identifies the specified models and checks them for compatibility. If all the prerequisites for the composition are fulfilled then a new valid model is returned to the user/stored in the repository.

An explicit composition scenario occurs when a consumer wants to make the best out of several heterogeneous models in DeVisa complying to the same mining schema. In this case it provides a *composition goal* $(MSpec, R)$, where *MSpec* is defined as above. R is a *validation set* with classified instances defined on the same schema as the models satisfying *MSpec*.

DeVisa uses a stacking approach (see 1) to train a meta-learner φ based on the outcomes of the existing DeVisa base models, i.e the models that satisfy *MSpec* and the relation R provided by the consumer. Let's assume that the base schema is $S = (U,P)$, where $U = \{A_1, A_2, \ldots, A_n\}$ and the base classifiers $\phi_1, \phi_2, \ldots, \phi_m : dom(A_1) \times dom(A_2) \times \cdots \times dom(A_n) \rightarrow dom(C)$, where $C \in U$. The meta-learner is a simple decision tree $\varphi : dom(C)^m \rightarrow dom(C)$ (by default DeVisa uses ID3), since the main work is done by the base classifiers. The outcome is another classifier $\phi : dom(A_1) \times dom(A_2) \times \cdots \times dom(A_n) \rightarrow dom(C)$, as depicted in Figure 2.

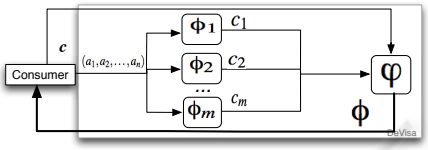

Figure 2: Explicit DeVisa composition - stacking approach.

This approach has the advantage of fitting the model to the consumer needs and the particularities of its own data (model customization instance).

Model composition is allowed only within a common data dictionary. Intuitively, a data dictionary refers to a strict domain. In the absence of a reference to a data dictionary in the consumer's goal, the same results can be achieved upon the availability of a common domain ontology, composed of concepts describing the domain in different abstraction levels, into which URLs are mapped.

3 ARCHITECTURE AND IMPLEMENTATION

3.1 PMQL

DeVisa defines a XML-based language called PMQL (Predictive Model Query Language) (PMQL, 2008) that is used to realize both the communication with the consumer application and the internal communication between DeVisa components. The consumer application expresses its goal in PMQL and wraps it in a SOAP ((SOAP, 2007)) message that is sent to the PMQL Web Service. The PMQL Web Service forwards the PMQL goal to a DeVisa component called *PMQL engine*, which is responsible with processing PMQL. It transforms the goal (query) so that it matches the existing DeVisa resources and, after successful matching, transfers the PMQL answer back to the consumer (Figure 3). To resolve a scoring or composition goal, the PMQL engine performs a sequence of steps: *annotation*, *rewriting* and *plan building* (Gorea, 2008). If a composition is necessary then an additional *composing* phase, which assembles the base models, is performed. The *execution* invokes certain XQuery (XQuery, 2007) functions against the repository of PMML models.

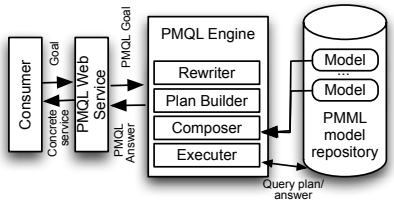

Figure 3: Resolving consumers' goals in DeVisa.

3.2 The PMQL Web Service

DeVisa's approach uses several layers of abstraction in defining its services. The PMQL Web Service is an abstract computational entity meant to provide access to the concrete DeVisa services. The materialization of those services is achieved after interpretation of the PMQL goal. The services are the effective values that are provided to the user as a response to its goal. They are tightly coupled with the business context (metadata) or with the circumstances of the request. More exactly, it depends on a data dictionary, on the existence of a DeVisa applicable model etc. The PMQL Web Service deals with arbitrary XML of the incoming and outgoing SOAP Envelopes without any type mapping / data binding - *message service* (Axis, 2007). The raw XML in a received SOAP envelope is passed to the PMQL engine, which attempts to interpret the XML as a PMQL query. This type of service has the advantage of separating the expression of the consumer's goal from the choreography of the concrete services.

DeVisa borrows some principles used in the WSMO framework (Fensel et al., 2007): web compliance, ontology as data models, strict decoupling of resources definitions, separation of the description from the implementation, ontological role separation, execution semantics. A clear distinction between a service and a web service is made. The web service is a computational entity able to achieve the user's goal by invocation whereas the service is the actual value provided by the invocation of the web service.The model composition in DeVisa follows a resembling pattern. The consumer provides a goal, providing the available input and the expected output. The composer will attempt to produce an orchestration, which at least produces all expected outputs, and at most expects all possible input messages.

4 CONCLUSIONS

The paper presented a theoretical foundation on the prediction models composition problem within an online scoring system called DeVisa. The prediction models are stored in PMML format in a native XML database. The model composition in DeVisa is limited by the non-availability of the original data set and to model sequencing and selection supported by PMML. Nevertheless, the consumer can provide a validation data set to train a new customized model. The paper identifies the contexts in which model composition can occur (implicit or explicit) and analyzes the possible approaches.To achieve a clear separation between the consumer's goal and the effective DeVisa services a specialized language (PMQL) is introduced. Because the main DeVisa functionality is available through Web Services, the model composition can follow some of the principles used in the web services composition process.

REFERENCES

Axis (2007). Apache axis. http://ws.apache.org/axis/.

DeVisa (2007). Devisa. http://devisa.sourceforge.net.

Fensel, D., Lausen, H., Polleres, A., de Bruijn, J., Stollberg, M., Roman, D., and Domingue, J. (2007). *Enabling Semantic Web Services*. Springer, Berlin Heidelberg.

Gorea, D. (2008). Devisa concepts and architecture of a data mining models scoring and management web system. In *Proceedings of the 10th International Conference on Enterprise Information Systems*. INSTICC.

Ian H. Witten, E. F. (2005). *Data Mining Practical Machine Learning Tools and Techniques*. Morgan Kaufmann series in data management systems. Elsevier.

Kuncheva, L. I. (2004). Classifier ensembles for changing environments. In *Proc. 5th Int. Workshop on Multiple Classifier Systems*, volume 3077 of *LNCS*, pages 1–15. Springer-Verlag.

M.Harries, Sammut, C., and Horn, K. (1998). Extracting hidden context. *Machine Learning*, 36(2):101–126.

PMML (2007). Pmml version 3.2. http://www.dmg.org/pmml-v3-2.html.

PMQL (2008). Predictive modelling query language. http://devisa.sourceforge.net/pmql.shtml.

Prodromidis, A., Chan, P., and Stolfo, S. (2000). *Meta-learning in distributed data mining systems: Issues and approaches*, chapter 3. AAAI/MIT Press.

SOAP (2007). Soap version 1.2 part 1: Messaging framework (second edition). http://www.w3.org/TR/soap12-part1/.

XQuery (2007). Xquery 1.0: An xml query language. http://www.w3.org/TR/xquery/.