USAGE OF ONTOLOGIES IN SYSTEMS OF DATA EXCHANGE

ONTOLOĢIJAS LIETOŠANA DATU MAIŅU SISTEMĀS

P.Osipovs and A.Borisov

Keywords: Machine Learning, ontologies, semantic WEB, OWL, RDF, software agents, Attribute Value Taxonomies

В статье рассмотрены методы и технологии, используемые для эффективного извлечения знания из больших массивов разнородных данных. Также рассмотрены методы структуризации сырых данных путём автоматической классификации с использованием онтологий.

В первой части статьи рассмотрены основные технологии, позволяющие реализовать Semantic WEB. Большое внимание уделено онтологиям, как важнейшему понятию, позволяющему структурировать информацию на очень высоком уровне.

Вторая часть рассматривает алгоритм AVT-DTL предложенный Jun Zhang, который позволяет автоматически создавать классификаторы по имеющимся сырым, потенциально неполным данным. Рассмотренный алгоритм использует новое понятие нефиксированных уровней онтологии и по результатам тестов превосходит лучшие существующие алгоритмы создания классификаторов.

Введение

Сложность структуры современного информационного общества постоянно растёт. В связи с этим, требования к эффективности алгоритмов обработки информации также увеличиваются. Общая модель знания из сырых данных представлена на Рис. 1. В последнее время наиболее популярными направлениями в этой области являются Data Mining (DM), Knowledge Discovery in Databases (KDD) и Machine Learning (ML). Все они предоставляют теоретическую и методологическую базу для изучения, анализа и понимания огромных объёмов данных.

Однако этих методов не достаточно если сама структура данных будет настолько плохо пригодной для машинного анализа, как исторически сложилось на сегодняшний момент в Internet. Постоянно возникает потребность предоставить удобный интерфейс доступа к разнообразным данным, дать возможность рассматривать их с удобной точки зрения в таких разных сферах как био-информатика, Е-коммерция, и т.д.

Для решения данной проблемы предпринята глобальная инициатива реорганизации структуры данных Internet в целях преобразования его в Семантическую Паутину, предоставляющую возможности по эффективному поиску и анализу данных, как человеком, так и программным агентам. Важнейшей составной частью Семантической Паутины является использование онтологий, которые и предоставляют возможности по представлению одних данных в разном виде, относительно потребностей и квалификации их запросившего.

Важнейшим недостатком существующей структуры Internet является то, что он практически не использует стандартов представления данных удобных для понимания компьютером, а вся информация предназначена в первую очередь для восприятия человеком. К примеру, для того, чтобы получить время работы семейного врача, достаточно зайти на сайт поликлиники и найти его в списке всех практикующих врачей. Однако если это просто сделать человеку, то программному агенту в автоматическом режиме это практически невозможно, если только не создавать его с учётом жёсткой структуры конкретного сайта.

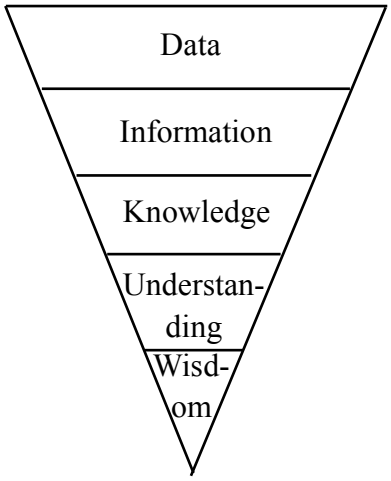

Рисунок 1 Процесс дистилляции знания

Для решения подобных проблем используются *онтологии*, позволяющие описать любую предметную область в понятных для машины терминах и эффективно использовать *мобильных агентов*.

При использовании такого подхода, дополнительно к видимой человеком информации на каждой странице имеются также и служебные данные, позволяющие эффективно использовать данные программными агентами.

В свою очередь онтологии являются составной частью глобального видения развития сети Internet на новый уровень, называемый **Semantic WEB** (SW) [1].

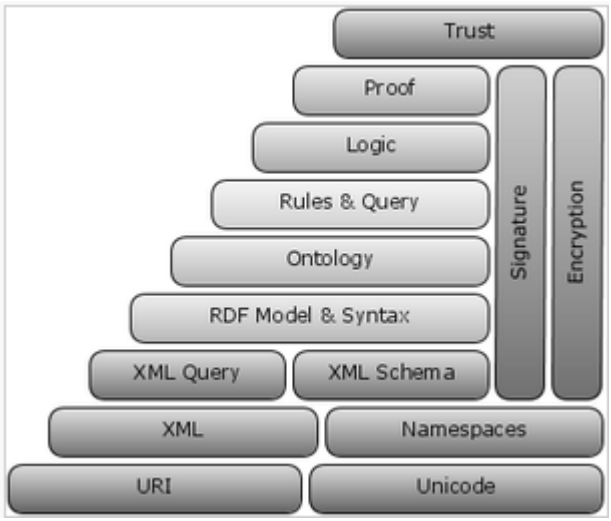

Рисунок 2 Стек понятий семантической паутины

Важнейшие понятия Semantic WEB

Для достижения столь сложной цели как глобальная реорганизация всемирной сети требуется целый набор взаимосвязанных технологий. На Рис.2 приводится общая структура понятий Semantic WEB. Ниже приводится краткое описание ключевых технологий.

Semantic WEB

Понятие семантической паутины является центральным в современном понимании эволюции Internet. Считается, что в будущем данные в сети будут представлены как в обычном виде страниц, так и в виде метаданных, примерно в одинаковой пропорции, что позволит машинам использовать их для логических заключений реализуя все преимущества от использования методов ML. Повсеместно будут использоваться *унифицированные идентификаторы ресурсов* (*URI*) и *онтологии*.

Однако, не всё так радужно, существуют и сомнения в возможности полной реализации семантической паутины. Основные тезисы в пользу сомнения в возможности создания эффективной семантической паутины:

- Человеческий фактор [2] люди могут врать, ленится добавлять метаописания, использовать неполные или просто неправильные метаданные. Как вариант решения данной проблемы можно использовать автоматизированные средства создания и редактирования метаданных.
- Излишнее дублирование информации, когда каждый документ должен иметь полное описание как для человека так и для машины. Это отчасти решается внедрением *микроформатов* [3].

Кроме самих метаданных, важнейшей составной частью SW является *семантические Web сервисы*. Они являются источниками данных для агентов семантической паутины, изначально нацелены на взаимодействие с машинами, имеют средства рекламы своих возможностей.

URI (Uniform Resource Identifier)

URI является унифицированным идентификатором любого ресурса. Может указывать как на виртуальный так и на физический объект. Представляет собой уникальную символьную строку общая структура которой представлена на Рис.3. Самым известным URI на сегодня является URL, являющейся идентификатором ресурса в Internet и дополнительно содержащий информацию о местонахождении адресуемого ресурса.

Рисунок 3 Базовый формат URI.

Онтологии

Применительно к области Machine Learning под онтологией понимается некая структура, концептуальная схема, описывающая (формализующая) значения элементов некоторой *предметной области* (**ПРО**). Онтология состоит из набора терминов и правил описывающих их связи, отношения.

Обычно онтологии строятся из *экземпляров*, *понятий атрибутов* и *отношений*.

- *Экземпляр* - элементы самого нижнего уровня. Главной целью онтологий является именно классификация экземпляров, и хотя их наличие в онтологии не обязательно, но как правило они присутствуют. **Пример**: слова, породы зверей, звёзды.
- *Понятия* - абстрактные наборы, коллекции объектов.

Пример: Понятие “звёзды”, вложенное понятие “солнце”. Чем является “солнце”, вложенным

понятием или экземпляром (светилом) – зависит от онтологии.

- Понятие “светило”, экземпляр “солнце”.
- *Атрибуты* - каждый объект может иметь необязательный набор атрибутов позволяющий хранить специфичную информацию.

Пример: объект “солнце” имеет такие атрибуты как:

- *Тип*: жёлтый карлик;
- *Масса*: $1.989 \cdot 10^{30}$ кг;
- *Радиус*: 695 990 км.

- *Отношения* - позволяют задать зависимости между объектами онтологии.

Так - как между различными онтологиями возможно установление точек пересечения, то использование онтологий позволяет смотреть на одну ПРО с различных точек зрения и в зависимости от задачи пользоваться различным уровнями детализации рассматриваемой ПРО. Понятие уровней детализации онтологии является одним из ключевых, к примеру, для обозначения цвета сигнала светофора иногда достаточно просто указать “зелёный”, тогда как при описании цвета покраски машины может не хватить даже такого детального описания как “*тёмно зелёный, близкий по тональности к хвое*”.
Рассмотрим общую структуру применения онтологий.

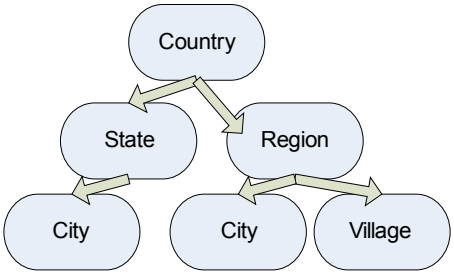

Рисунок 4 Часть возможной онтологии адресов

В случае использования онтологии приведённой на Рис.4, для того, чтобы отослать письмо в американский университет, достаточно указать его название, программный агент сам найдёт его адрес на основе стандартной адресной информации с сайта университета, если нужно отослать письмо на конкретный факультет, то с сайта будет получен список всех факультетов и выбран нужный, и уже с сайта требуемого факультета взят адрес, далее, используя вышеприведённую онтологию программа определит формат адреса принятый в США.
Компьютер не понимает всей информации в полном смысле слова, но использование онтологий позволяет ему намного более эффективно и осмысленно пользоваться доступными данными.

Конечно, остаётся много вопросов, к примеру, как в начале агент найдёт сайт требуемого университета? Однако для этого уже сейчас разработаны средства. К примеру, Язык Онтологии Сетевых Сервисов (Web Services Ontology Language, OWL-S [4]) который позволяет сервисам рекламировать свои возможности, услуги.

На Рис. 5 приведён вариант разделения онтологий по уровням формализации отображающим степень онтологической точности каждого метода.

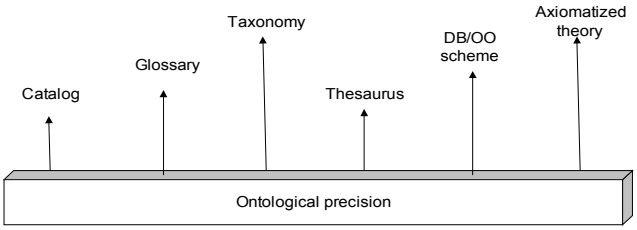

Рисунок 5 Иерархия таксономий

Под онтологической точностью понимается возможность онтологии различать оттенки значений.

- *Каталог* – содержит набор нормализованных терминов, не имеет иерархической структуры.
- *Глоссарий* – все термины упорядочены по алфавиту естественного языка.
- *Таксономия* – частично упорядоченная коллекция концептов (таксонов) разбивающая предметную область от общих к конкретным понятиям.
- *Аксиоматизированная таксономия* – подвид таксономий, содержит аксиомы определённые на формальном языке.
- *Контекстные библиотеки / аксиоматические онтологии* – являются взаимосвязанными наборами аксиоматизированных таксономий. Представляют собой пересечение нескольких контекстов, или включение одного контекста в другой.

Таксономии

Таксономии являются одним из вариантов реализации онтологий. С помощью таксономии возможно определить классы, на которые делятся объекты некоторой предметной области, а также то, какие отношения существуют между этими классами. В отличие от онтологий, задача

таксономий чётко определена в рамках иерархической классификации объектов.

Таксономии Значений Атрибутов

Важным примером использования понятия Таксономии в Machine Learning является понятие *Таксономии Значений Атрибутов* (Attribute Value Taxonomies (AVT).
Формальное определение AVT. Пусть имеется вектор атрибутов $A = \{A_1, A_2, ..., A_N\}$, а $dom(A_i)$ определяет набор значений атрибута A_i. В таком случае, формальное определение AVT выглядит следующим образом:
Таксономия Значений Атрибутов T_i атрибута A_i это древовидная структура, содержащая частично упорядоченную иерархию концептов представленную в виде множества $(dom(A_i), \square)$, где $dom(A_i)$ это конечное множество, содержащее все атрибуты в A_i, и $\square$ признак частичного порядка определяющего отношения среди значений атрибутов в $dom(A_i)$. Совместно, $T = \{T_1, T_2, ..., T_N\}$ представляют упорядоченное множество таксономий значений атрибутов вязанных с $A_1, A_2, ..., A_N$.
Пусть $Nodes(T_i)$ представляет собой набор всех значений T_i, а $Root(T_i)$ указывает на корневой элемент таксономии T_i. Множество листьев $Leaves(T_i)$ представляют все примитивные значения атрибутов A_i. Тогда, абстрактными значениями атрибутов A_i будет называться дерево $Nodes(T_i) - Leaves(T_i)$. А каждое ребро дерева представляет собой отношение ***is-a*** между атрибутами всей таксономии. Таким образом AVT определяет абстрактную иерархию между значениями атрибутов.

Современные языки описания онтологий

RDF (Resource Description Framework) [5] язык описания метаданных ресурсов, главной его целью является представление утверждений в виде одинаково хорошо воспринимаемом как человеком, так и машиной.
Атомарным объектом в RDF является триплет: субъект – предикат – объект. Считается, что любой объект, можно описать в терминах простых свойств и значений этих свойств.

Table 1
Пример таблицы с выделенными параметрами

:me	:owns	:my_appartment
:me	:owns	:my_computer
:me	:is_in	:philadelhia
:my_appartment	:has_a	:my_computer
:my_appartment	:has_a	:my_bed
:my_appartment	:is_next_to	:my_bed

Перед двоеточием должен указываться Уникальный Идентификатор Ресурса URI (Uniform Resource Identifier), однако в целях экономии трафика допускается указать только пространство имён.

Дополнительно, в целях улучшения восприятия человеком, существует практика представления схем RDF в виде графов.

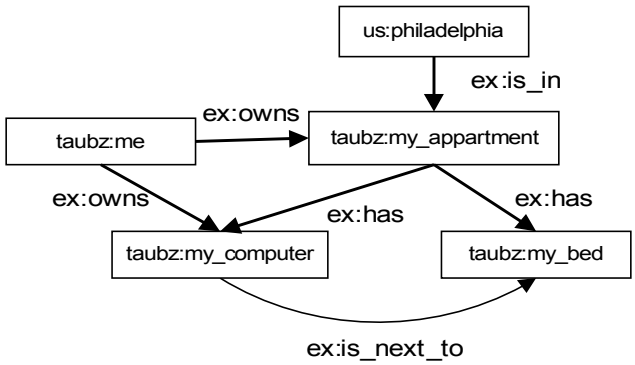

Рисунок 6 Пример представления схемы RDF в виде графа

На Рис. 6 приведён пример представления структуры RDF в виде графа. Рёбра представляют собой утверждения, имя в начале ребра - подлежащее утверждения, в конце его дополнение, имя у самой дуги – сказуемое.

OWL (Web Ontology Language)[6] язык Веб онтологий, созданный для представления значения терминов и отношения между этими терминами в словарях. В отличии от RDF, данный язык использует более высокий уровень абстракции, что позволяет языку наряду с формальной семантикой использовать дополнительный терминологический словарь.

Важным преимуществом OWL является то, что его основу положена чёткая математическая модель дескрипционных логик [7].

Место OWL в стеке понятий Semantic WEB представлено на Рис. 7.

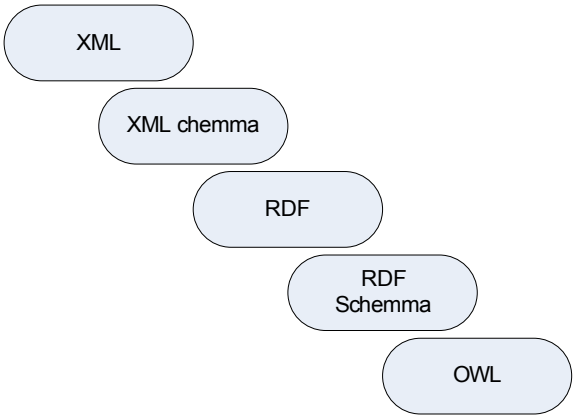

Рисунок 7 Место OWL в общей структуре Semantic WEB с точки зрения консорциума W3C

- XML – предоставляет возможности создания структурированных документов, но не предъявляет к ним никаких семантических требований;
- XML Schema – определяет структуру XML документов и дополнительно позволяет использовать конкретные типы данных;
- RDF – предоставляет возможность описывать абстрактные модели данных некоторых объектов и отношения между ними. Использует простую семантику на основе XML синтаксиса;
- RDF Schema – позволяет описывать свойства и классы RDF – ресурсов, а также семантику отношений между ними;
- OWL – расширяет описательные возможности предыдущих технологий. Позволяет описывать отношения между классами (к примеру непересекаемость), кардинальность (например “точно один”), симметрия, равенство, перечисляемые типы классов.

По степени выразительности выделяют три диалекта OWL

- *OWL Lite* – является подмножеством полной спецификации, предоставляющим минимально достаточные средства описания онтологий. Предназначен для снижения первичного внедрения OWL. А также для упрощения миграции на OWL тезаурусов и прочих таксономий. Гарантируется, что логический вывод на метаданных с выразительностью OWL Lite выполняется за полиномиальное время (сложность алгоритма принадлежит классу P).

  Диалект основан на дескрипционной логике $SHLF(D)$.
- *OWL DL* – с одной стороны предоставляет максимальную выразительность, полноту вычислений (все они будут гарантированно вычисляемыми) и полную разрешаемость (все вычисления завершаются в определённое время). Но в связи с этим имеет строгие ограничения, к примеру на взаимосвязи классов и время выполнения некоторых запросов по таким данным могут требовать экспоненциального времени выполнения.

  Диалект основан на дескрипционной логике $SHOLN(D)$.
- *OWL Full* – предоставляет максимальную выразительную свободу, но не даёт никаких гарантий разрешаемости. Все созданные структуры опираются обоснованы только реализуемым алгоритмом. Считается маловероятным, что какое-либо рассудочное программное обеспечение будет в состоянии поддержать полную поддержку каждой особенности OWL Full.

  Не соответствует ни одной дескрипционной логике, так – как в принципе является не разрешимым.

На данный момент язык OWL является основным инструментом описания онтологий.

Программные (мобильные, пользовательские) агенты (SA)

В рассматриваемой ПРО SA считается программой, действующей от имени пользователя, самостоятельно выполняющей сбор информации на протяжении некоторого, возможно длительного времени находясь вне полностью предсказуемого окружения. Также важной их особенностью является возможность взаимодействовать и сотрудничать с другими агентами и сервисами для достижения поставленной цели.

В отличие от ботов поисковых машин, которые просто сканируют диапазоны WEB страниц, агенты перемещаются от сервера к серверу, т.е. на отправном сервере агент уничтожается, а на принимающем создаётся с полным набором собранной ранее информации. Такая модель позволяет агенту использовать доступные серверу, источники данных, которые не доступны посредством WEB интерфейса.

Понятно, что на сервере должна быть установлена платформа, позволяющая принять агента и обслужить его запросы. Также важно уделить внимание безопасности и целостности агентов. Для этого применяется подход выделенных

пространств, когда агент работает в некотором безопасном окружении с ограниченными правами и возможностями воздействия на систему.
По своей реализации агенты обычно делятся на две категории по уровню предоставляемой им свободы:

- *Nointelligent agents* – по сути своей являются подобием функции с чётко заданным количеством параметров. Он жёстко нацелен на выполнение определённой задачи в строго определённых условиях.
- *Intelligent agents* – могут сами выбирать технологию которую нужно использовать в каждом случае для решения задачи. Часто подобные программы реализуются с помощью нейронных сетей, которые позволяют агенту постоянно подстраиваться под меняющиеся условия.

Микроформаты

Микроформаты являются попыткой создать семантическую разметку разнообразных сущностей на Web-страницах одинаково хорошо воспринимаемую как человеком так и машиной. Информация в некотором микроформате не требует применения дополнительных технологий или пространств имён дополнительно к простому (X)HTML. Спецификация микроформата, это просто соглашение на стандарты поименования классов элементов оформления страницы позволяющих хранить в каждом из них соответствующие данные.
Для иллюстрации разберём формат hCalendar.
Данный микроформат является подмножеством формата iCalendar (RFC 2445) и предназначен для описания дат будущих или прошедших событий для предоставления возможностей их автоматической агрегации поисковыми агентами.

```
<div class="vevent">
  <a class="url" href="http://www.web2con.com/">
    http://www.web2con.com/
  </a>
  <span class="summary">
    Web 2.0 Conference
  </span>:
  <abbr class="dtstart" title="2007-10-05">
    October 5
  </abbr>
  -
  <abbr class="dtend" title="2007-10-20">
    19
  </abbr>
  ,at the
  <span class="location">
    Argent Hotel, San Francisco, CA
  </span>
</div>
```

В данном примере приведено описание создания корневого класса контейнера с датой (class="vevent") и соотнесение с событием некоей даты в стандартном формате ISO date.

На данный момент самыми распространёнными микроформатами являются

- hAtom - формат рассылок новостей;
- hCalendar - составление календаря и описание событий;
- hCard - описание людей, компаний, мест;
- hResume - формат описания резюме;
- hReview - внедрение обзоров;
- XFN - способ указания отношений между людьми;

Современные направления развития онтологий

Составление онтологий с использованием экспертного подхода является очень медленным и дорогим процессом, и хотя результат может быть очень хорошим, но в подавляющем большинстве используют автоматически классификаторы, которые строят онтологии по имеющимся данным. На этом пути множество проблемных мест, и поэтому существующие автоматические классификаторы недостаточно качественны и одним из вариантов повышения точности классификации является *Ontology Aware Classifiers* [9].

При использовании автоматических методов добычи данных (data-driven knowledge discovery) иногда возникает потребность исследовать полученную информацию с различных точек зрения. Особенно это важно в научных исследованиях, когда результат анализа данных сильно зависит от использованной для их представления онтологии. В частности, это один из важнейших моментов в процедуре автоматического обучения классификаторов данных на основе их онтологий. К примеру, если анализируемые данные в разных частях представлены атрибутами с разной точностью (что очень часто встречается в реальных задачах, к примеру, если данные изначально собираются из различных источников, с разной терминологией), то на сегодняшний момент нет алгоритма, который бы учитывал это и использовал

разные уровни онтологии для различных частей данных и эффективно работал с потенциально недостаточно полно заданными данными.

Следующие требования являются предпосылками для создания нового поколения классификаторов:

- требуются более простые, мощные и робастые классификаторы;
- классификаторы использующие абстрактные значения атрибутов зачастую предоставляют наиболее простые модели разделения данных;
- в случае недостаточных данных оценка полученная на основе абстрактных значений атрибутов зачастую более надёжна, чем на основе примитивных значений.

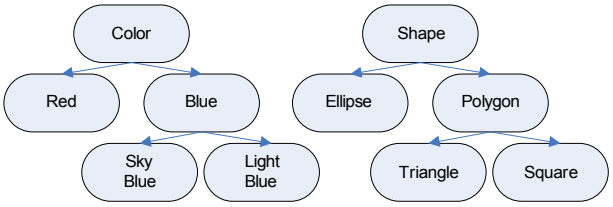

Рисунок 8 Часть онтологий цвета и описания фигур

На Рис. 8 можно видеть примеры частично и полностью определённых фигур. К примеру, частично определены такие данные как {blue, ellipse}, {sky blue, triangle}, полностью определены {light blue, square}.

Существует метод создания обучающегося классификатора шаблонов на основе Таксономии Значений Атрибутов (Attribute Value Taxonomies (AVT) при наличии потенциально недостаточных данных. Алгоритм использует деревья принятия решений и называется (**AVT-DTL** Attribute Value Taxonomies - Decision Tree Learning algorithm).

Имеется онтология цветов (Рис. 8), а в анализируемых данных некоторые атрибуты описаны как голубые, некоторый как красные, а некоторые как небесно-голубые. В таком случае алгоритм производящий классификацию по нескольким уровням онтологии будет более точным.

Классический метод построения дерева классификации на каждом шагу выбирает наиболее информативный атрибут и объявляет его узлом на текущей итерации. В отличие от него AVT-DTL на каждой итерации выбирает не просто атрибут, а уровень таксономии, на котором он представлен. Этот уровень представляется в виде вектора номеров уровней во всех участвующих в классификации таксономий. Далее, на основе всех созданных таким образом векторов создаётся нижняя граница, по минимальным значениям векторов для каждой таксономии. В заключении производится проверка на достаточность информативности полученных ограничений.

Определимся с терминами

- AVT(A) – таксономия атрибута A;
- Leavs(AVT(A)) – множество всех возможных значений A;
- все внутренние, промежуточные узлы называются абстрактными;
- разрезом называется условная линия абстракции таксономии, к примеру {красный, голубой}, {красный, небесно голубой, бирюзовый}.

$$\sigma_i(d,S) \leftarrow \sigma_i(d,S)\left(1+\frac{1}{\sum_{d \in Children(v,T_i)} \sigma_i(d,S)}\right)$$

Для достижения каждого листа, существует один путь T, который можно представить как множество путей до всех вышестоящих узлов $T = \{T_1, T_2, .. T_i\}$, где $T_i = AVT(A_i)$.

Также дополнительно используются следующие операции на таксономии T_i ассоциированной с A_i атрибутом:

- *prim* (Ti)=*Leaves*(AVT(Ai)); количество узлов i-ой таксономии;
- *depth*(Ti , $v(Ai)$); длинна пути от корня до значения i-ого атрибута таксономии;
- *leaf*(Ti , $v(Ai)$); булевское значение, правда если i-ый атрибут в таксономии Ti является листом, т.е. $v(Ai) \in Leaves(Ti)$.

Считается, что атрибут полностью определён, если он имеет значение одного из листов таксономии $A \in Leaves(AVT(A))$, в противном случае атрибут определён частично.

Данные объявляются полностью определёнными, если каждый атрибут имеет максимально полное описание: $\forall i \; v_i^{(j)} \in prim(T_i)$.

Рассмотренный алгоритм является реализацией поиска с верху вниз по пространству гипотез построения дерева решений. Алгоритм отдаёт предпочтение разделению по как можно более абстрактным (близким к корню) атрибутам. Это позволяет для каждого атрибута выбирать уровень его абстракции в таксономии.

В описании алгоритма используются следующие определения:

- $A=\{A_1, A_2, ..., A_n\}$ упорядоченное множество атрибутов.
- $T=\{T_1, T_2, ..., T_n\}$ множество таксономий.

- $C=\{C_1, C_2, ..., C_m\}$ множество целевых, взаимно не пересекающихся классов.
- $\psi(v, T_i)$ - множество потомков узла соответствующих значению v в таксономии T_i.
- *Children* (v, T_i) - множество потомков узла v в таксономии T_i.
- $\Lambda(v, T_i)$ – список предков, включая корень, для v *в* T_i.
- $\sigma_i(v, S)$ – количество вхождений значения v атрибута A_i в тренировочное множество S.
- *Counts*(T_i) – дерево, узлы которого содержат цифровые значения атрибутов в таксономии T_i.
- $P_s = \{p_1^{(s)}, p_2^{(s)}, .., p_n^{(s)}\}$ - вектор значений (ещё называемый границей AVT), состоящий из значений узлов множества S, где узел $P_i^{(s)}$ указывает на значения узла T_i атрибута A_i.

$\Phi(P_s)$ = true, только в случае если $\forall p_i^{(S)} \in P_s$, и $\Psi(p_i^{(S)}, T_i) = \{\}$.

Алгоритм AVT-DTL работает по направлению от корня каждого AVT дерева к листьям и строит деревья решений, которые используют наиболее абстрактные атрибуты, которые достаточно информативны для классификации тренировочного множества, потенциально содержащего недостаточные данные. В процессе своей работы алгоритм на, основе имеющихся AVT, производит поиск алгоритмом hill-climbing (в котором каждый следующий узел выбирается по мере увеличения его близости к решению) по пространству гипотез о вариантах построения дерева решений.

Алгоритм состоит из двух основных шагов:

1. Вычисление весов на основе имеющейся AVT.
2. Построение дерева решений на основе полученных весов.

Подробнее о каждом из шагов.

1. Вычисление весов на основе имеющейся AVT состоит из следующих операций:
   a. Создание корневого элемента, задание тренировочного множества S, заполнение множества $P_S=\{A_1, A_2, ..., A_n\}$ так, чтобы каждый из элементов данного вектора использовал значение соответствующей таксономии $T_1, T_2, ..., T_n$.
   b. Инициализация вычисления весов на основе анализа тренировочных данных;
      i. Аккумулирование весов ассоциированных с каждым значением каждого атрибута тренировочного множества. Таким образом для каждого $T_i \in T$ вычисляется $\sigma_i(v, S)$ для каждой таксономии.
      ii. Для каждого $T_i \in T$ обновляются веса всех родительских элементов в случае если на предыдущем шаге для i-го узла было получено не нулевое значение. В результате получается дерево *Counts*(T_i).
      iii. Для каждого $T_i \in T$ и для всех неполных атрибутов значение v в каждом случае $Ij \in S$, рекурсивно высчитывает значения весов для всех потомков v в таксономии T_i на основе *Counts*(T_i) и обновляет *Counts*(T_i). Таким образом, для каждого d из $\psi(v, T_i)$, обновляются $\sigma_i(d, S)$ по следующей формуле:

(1)

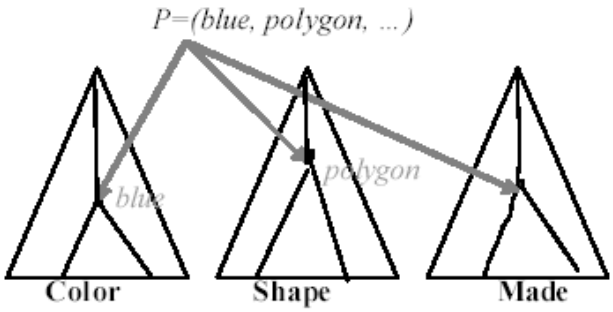

Рисунок 9 Образец вектора указателя P

В качестве иллюстрации, на Рис. 8 изображён вектор указывающий на два значения атрибутов *blue* и *polygon* у двух таксономий для применено следующее правило: If (*Color=blue & Shape=polygon &...*) then *Class* =+.

2. Построение дерева решений на основе полученных весов состоит из следующих операций ($Tree - Build(S, P_S)$:
   a. Если каждый элемент из тренировочного множества S помечен как C_i, то возвращается значение C_i, в противном случае если $\Phi(P_s) = true$ то отмечается потребность в создании конечного (листового) элемента (Критерий Останова Stopping Criterion).
   b. Для каждого указателя $P_i^{(S)}$ в векторе P_s делать:
      i. Проверить каждое частично неопределённое значение с частично определённым атрибутом v на атрибут A_i с

указателем $P_i^{(S)}$. Если $depth(Ti\ ,\ P_i^{(S)}) < depth(Ti\ ,\ v)$, то частично неопределённое значение трактуется как полностью определённое и отправляется в соответствующий уровень дерева решений с корнем $P_i^{(S)}$. В противном случае заменяет v с вероятностью d (элемент $Children\ (v,\ T_i)$) в соответствии с распределением весов связанных с $Children\ (v,\ T_i)$ по следующему правилу:

$$\Pr(d) = \frac{\sigma(d, T_i)}{\sum_{e \in Children(v, T_i)} \sigma(e, T_i)} \quad (2)$$

ii. Вычисляется энтропия множества S на основе разделения его по $Children\ (p_i^{(S)},\ T_i)$.

c. Выбирается лучший указатель $P_a^{(S)}$ в P_S для того чтобы, данное разделение S обеспечило максимальный прирост информации.

d. Разделение множества S на подмножества $S_1, S_2, .., S_{|Children(P_a^{(S)}, T_i)|}$ используя данные из $Children\ (p_a^{(S)},\ T_i)$.

e. Расширение P_S для получения нового указывающего вектора соответствующего подмножеству $S_1, S_2, .., S_{|Children(P_a^{(S)}, T_i)|}$ путём изменения указателя $Pa^{(S)}$ на значение атрибута A_i из соответствующего подмножества S_j $(1 \le j \le |Children(p_a^{(S)}, T_i)|)$.

f. Для каждого $(1 \le j \le |Children(p_a^{(S)}, T_i)|)$ проделать $Tree - Build(S_j, P_{s_j})$.

Описание оценки алгоритма

Для подтверждения эффективности данного алгоритма было проведено сравнительное тестирование на одной и той же выборке с другим, наиболее популярным на сегодняшний момент алгоритмом C 4.5.

Результаты доказали эффективность данного алгоритма, особенно важным стало заметное увеличение качества итоговых классификаторов построенных по выборке с большим процентом missing data. Итоговый выигрыш находится в пределах 5%-30% увеличения эффективности классификации. Также отмечено, что созданные алгоритмом AVT-DTL классифицирующие деревья содержат меньше узлов и более компактны, что позволит получить выигрыш в объёмах ресурсов требуемых для их использования.

Заключение

На дынный момент, трудно прогнозировать какими будут системы обмена данными в будущем, однако направления развития закладываются уже сейчас и от их эффективности многое зависит.

В статье рассмотрены основные технологии позволяющее создавать сложные системы, которые могут быть эффективно использованы как людьми, так и машинами. В стек понятий Semantic WEB на разных уровнях входит большое количество различных технологий, но только совместное их применение позволит добиться синергетического эффекта.

Так – как эффективность технологий каждого уровня стека влияет на всю систему, то рассмотренный во второй части работы алгоритм **AVT-DTL** улучшая качество автоматических классификаторов в итоге увеличит эффективность всей Семантической Паутины.

Об авторах

Pavel Osipov Mg.Sc.Eng. Ph.d. student, Institute of Information Technology, Riga Technical University. He received his masters diploma in Transport and Telecommunications Institute, Riga. His research interests include web data mining, machine learning and knowledge extraction.

Arkady Borisov, Dr.habil.sc.comp.,Professor, Institute of Information Technology, Riga Technical University, 1 Kalku Street, Riga LV-1658 Latvija, e-mail: arkadijs.borisovs@cs.rtu.lv.

References

1. Main page of W3 consortium, Semantic Web ideology creator http://www.w3.org/2001/sw/.
2. Cory Doctorow Metacrap: Putting the torch to seven straw-men of the meta-utopia http://www.well.com/~doctorow/metacrap.htm.

3. Main microformats ideology site http://microformats.org/.
4. Main page of W3 consortium, Web Ontology Language for Services http://www.w3.org/Submission/2004/07/.
5. Mai page of W3 consortium, Resource Description Framework http://www.w3.org/Submission/2004/07/.
6. Main page of W3 consortium, Web Ontology Language (OWL) http://www.w3.org/2004/OWL/.
7. Baader F., Horrocks I., Sattler U., Description Logics as Ontology Languages for the Semantic Web // Berlin: Springer ISSN 0302-9743, February 2005. - P. 228-248.
8. Main page of microformats community http://microformats.org/.
9. Zhang J., Learning ontology aware classifiers // Dissertation of the requirements for the degree of DOCTOR OF PHILOSOPHY / Iowa State University 2005

Pāvels Osipovs, Arkadijs Borisovs. Ontoloģijas lietošana datu maiņu sistemās

Pantā aprakstītas metodes un paņēmienus, ko izmanto, lai efektīvi iegūt zināšanas no liela apjoma jaukta datus. Arī jāmeklē metodes struktūras izejas datu automātisku klasifikāciju, izmantojot ontoloģijas.

Pantā pirmā daļa pamata apskatiti tehnoloģijas kuri ļaujot realizēt Semantiskā WEB. Liela uzmanība tiek veltīta ontoloģiju, kā galvenie jēdzieni, kadi nodrošina apkopotu informāciju, uz ļoti augsta līmeņa.

Otrajā daļā pārbauda AVT-DTL algoritmu ieteica Jun Zhang, kas ļauj automātiski izveidot klasificētāji saskaņā ar pieejamo izejvielu, iespējams nepilnīgus datus. Uzskata algoritms izmanto jaunu koncepciju peldošā līmeņa ontoloģijas un par testu rezultātiem, pārsniedz labāko esošos algoritmus radītu klasifikatori.

Pāvels Osipovs, Arkadijs Borisovs. Usage of ontologies in systems of data exchange

This paper describes the methods and techniques used to effectively extract knowledge from large volumes of heterogeneous data. Also look at methods to structure the raw data by the automatic classification using ontology.

In the first part of the article the basic technologies to realize the Semantic WEB described. Much attention is paid to the ontology, as the major concepts that structure information on an very high level.

The second part examines AVT-DTL algorithm proposed by Jun Zhang, which allows automatically create classifiers according to the available raw, potentially incomplete data. The considered algorithm uses a new concept of floating levels of ontology and the results of the tests exceeds the best existing algorithms for creating classifiers.