Fachbereich Informatik

Klassifikation von Web Services

Bachelorarbeit

zur Erlangung des Grades eines
Bachelor of Science
im Studiengang
Informationsmanagement

vorgelegt von

Marc-André Meurer

Betreuer: Prof. Dr. Steffen Staab, Institut für Informatik
Christoph Ringelstein, Institut für Informatik

Erstgutachter: Prof. Dr. Steffen Staab, Institut für Informatik
Zweitgutachter: Christoph Ringelstein, Institut für Informatik

Koblenz, im Februar 2006

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.

Koblenz, im Februar 2006

_____

Einleitung

Wegen der wachsenden Zahl von Web Services untersuche ich im Rahmen meiner Bachelorarbeit den Einsatz verschiedener maschineller Lernverfahren des Data Mining zur Klassifikation in kennzeichnende Kategorien.
Zentrale Frage: *Kann man mit Hilfe einer Menge von klassifizierten WSDL-Dokumenten auch unbekannte WSDL-Dokumente in diese Klassen einordnen?*

Web Services sind die Implementierung einer Service-orientierten Architektur (SOA) und verfolgen das Ziel, die Kommunikation zwischen Maschinen automatisiert stattfinden zu lassen. Deshalb verwendet man im Bereich von Web Services auch den Begriff der „Maschine-zu-Maschine-Kommunikation“. Das Szenario einer Service-orientierten Architektur beschreibt das Anbieten, Suchen und Nutzen von Diensten über ein Netzwerk. Der Zugriff auf einen Webdienst wird nur über Schnittstellen erlaubt. Damit die Nutzung von Web Services für verschiedene Dienstnutzer möglichst einfach und komfortable gehalten wird, beschreibt ein Dienstanbieter die Schnittstellen und angebotenen Funktionalitäten seines Dienstes mit Hilfe einer Dienstbeschreibung. Im Rahmen von Web Services entwickelte man hierzu, die XML-basierte Beschreibungssprache *Web Service Description Language* (WSDL). WSDL-Dokumente enthalten eine standardisierte Struktur, mit der sich kennzeichnende Eigenschaften von Web Services beschreiben lassen. Damit bilden sie die Grundlage für meine Untersuchung mit verschiedenen Methoden und Verfahren des Data Mining, mit dem Ziel, aussagekräftige Muster und Zusammenhänge von Web Services zu entdecken.

Vor der eigentlichen Untersuchung mit Hilfe von maschinellen Lernverfahren des Data Mining, müssen die Daten aber zunächst gesammelt und entsprechend aufbereitet werden. Als Quelle für klassifizierte WSDL-Dokumente bieten sich die Datenquellen des ASSAM-WSDL-Annotator-Projekts an - diese enthalten neben den eigentlichen WSDL-Dokumenten noch Zusatzinformationen über Herkunft und Kategorie der Web Services. Um die Daten aus den einzelnen WSDL-Dokumenten zu extrahieren, programmiere ich in PHP, einen DOM-basierten Parser, der mir diese Arbeit abnimmt. Als Ergebnis erhalte ich eine Tabelle, in der sich die spezifischen Eigenschaften der Web Services und die Klassenzugehörigkeit finden.

Die Tabelle mit WSDL-Daten ist Grundlage für die Untersuchung mit der Data-Mining-Software WEKA, die verschiedene Werkzeuge zur Datenanalyse und –vorbereitung und eine Vielzahl implementierter Lernverfahren zur Verfügung stellt. Zur Modellierung der WSDL-Daten entwickele ich verschiedene Strategien und untersuche diese mit WEKA. Als Ergebnis meiner Bachelorarbeit präsentiere ich ein Verfahren, das unbekannte WSDL-Dokumente erfolgreich einer Klasse zuordnet.

Inhaltsverzeichnis

1.1 Web Services – Idee, Entwicklung und Anwendungsszenario

> *„A web service is a piece of business logic, located somewhere on the Internet, that is accessible through standard-based Internet protocols […] Using a web service could be as simple as logging into a site or as complex as facilitating a multi-organization business negotiation."*
> *(Chappell & Jewell, 2002, S.1)*

Die Idee von Web Services ist der automatische Austausch von Daten und Funktionalität zwischen Kommunikationspartnern. Im Unterschied zum bisherigen Web-Ansatz, bei dem ein Mensch als Nutzer bestimmte Daten und Funktionalität von einer Webseite abruft, verfolgen Web Services das Ziel, die Kommunikation zwischen Maschinen in Form von Anwendungen automatisch stattfinden zu lassen. Deshalb verwendet man im Bereich von Web Services auch den Begriff der „Maschine-zu-Maschine-Kommunikation".

Die Nutzung von Web Services ist unkompliziert: Zum Beispiel bietet die Suchmaschine Google[1] einen Webdienst an, um die Suchmechanismen auf der eigenen Webseite einzubinden. Als Antwort, auf eine erfolgreiche Suchanfrage, schickt Google die angeforderten Daten zu.
Die Nutzung von Web Services kann aber auch komplexe Probleme lösen: Zum Beispiel verwendet die Firma Dell in der Praxis Web Services in ihrem Warenwirtschaftssystem, um auch kleinere Zulieferer über das Internet, in das firmeneigene System anzubinden.
(vgl. Hauser & Löwer, 2004, S.12)

Um die Vorteile von Web Services zu verdeutlichen, betrachten wir den beispielhaften Einsatz von Web Services für ein Online-Reisebüro: Der Anbieter eines Online-Reisebüros möchte seinen Kunden die Möglichkeit bieten, die Daten seiner Lieferanten (Fluggesellschaften, Hotel- und Mietwagenanbieter) online aufzurufen und buchen zu können.

Eine Datenbank anzulegen, wo alle Daten seiner Lieferanten erfasst und gespeichert sind, ist die erste Variante, die der Anbieter des Online-Reisebüros verfolgen könnte. Jedoch muss die Datenbank konstant gepflegt und aktualisiert werden, was hohen personalen Aufwand erfordert.
Als zweite Variante könnte der Anbieter des Online-Reisebüros, für jede Kunden-Anfrage selbst die Webseiten seiner Lieferanten aufsuchen und das Ergebnis seiner Recherche an den Kunden zurückzugeben. Nachteilig wäre, dass abgesehen vom hohen Zeitaufwand für jede einzelne Kunden-Anfrage, das Online-Reisebüro nur eine begrenzte Anzahl von Kunden zur gleichen Zeit mit Informationen versorgen kann.
Eine automatisierte Recherche auf den Webseiten seiner Lieferanten ist eine effektive Alternative. Zunächst muß der Anbieter des Online-Reisebüros aber einige Probleme lösen: Für Menschen sind Webseiten leicht verständlich. Für Maschinen sind Webseiten weniger verständlich, da sie in der Regel nicht für automatisierte Zugriffe entworfen sind. Die

[1] http://www.google.de/api

Beschreibung des Dokumentenlayouts in HTML (Hypertext Markup Language) macht Webseiten für den Menschen zwar recht übersichtlich, Maschinen jedoch erkennen nicht direkt, welche Daten eine Webseite repräsentiert. Die Semantik der Daten, also Sinn und Bedeutung, gehen beim HTML-Layout verloren.
Das Problem könnte umgangen werden, indem man dem Online-Reisebüro einen direkten Zugriff auf die firmeneigenen Datenbanken die Lieferanten gewährt. Jedoch wird nicht jeder Lieferant einen direkten Zugriff auf seine Datenbanken erlauben - die wichtigsten Daten des Unternehmens sind dort abgelegt, eine Zerstörung oder Fehlfunktion hätte katastrophale wirtschaftliche Folgen.
Sollte dem Online-Reisebüro wegen seiner großen Vertrauenswürdigkeit und technischer Kompetenz doch ein direkter Zugriff auf die Datenbanken seiner Lieferanten erlaubt sein, gibt es das nächste Problem zu lösen. Es existiert eine Vielzahl von Möglichkeiten, auf die Schnittstellen von Datenbanken zuzugreifen. Besonders der Middleware-Ansatz bietet eine komfortable Schnittstellen-Programmierung, um über Netzwerk auf eine Datenbank zuzugreifen. Jedoch finden sich beim Middleware-Ansatz auch Nachteile, wie die Nutzung von relativ komplexen und oft proprietären Protokollen zur Übertragung, was die Dienste zueinander inkompatibel macht.

Als Konsequenz entwickelte man den recht neuen Ansatz der Web Services, um die Nachteile des Middleware-Ansatzes zu beseitigen. Da sich Web Services auf öffentlichen Webservern installieren lassen und die Kommunikation über sehr einfache und vor allem standardisierte Protokolle stattfindet, lassen sich Web Services sehr leicht überall ohne größere Kosten anbieten und nutzen.

Bei dem Szenario des Online-Reisebüros sollte der Anbieter über den Einsatz von Web Services nachdenken. Dazu müssen die Lieferanten entsprechende Web Services auf ihren Webservern zur Verfügung stellen, mit denen eine automatisierte Kommunikation stattfinden kann: Der Kunde sendet an die Webseite des Online-Reisebüros eine Anfrage, die automatisiert an die Lieferanten weitergeleitet, verarbeitet und als Ergebnis die angeforderten Daten zurückgibt.
(vgl. Eberhart & Fischer, 2003, S.2)

1.2 Web Services – Service-orientierte Architektur

> *„In a typical scenario, a provider hosts the implementation for a service. Providers define service descriptions for services and publish them to a registry. A requestor then uses a registry to find service descriptions for services they are interested in using. With the service description in hand, the requestor binds (i.e., creates a service request for) to a service."*
> *(Chappell, Jewell, 2002, S.15)*

Eine Service-orientierte Architektur (SOA) bezeichnet die Abstraktion einer konkreten Technik: Sie beschreibt ein Bild der Wirklichkeit, was wesentliche Aspekte hervorhebt und unwesentliche Aspekte vernachlässigt. Die aussichtsreichste Implementation dieser Abstraktion ist der Web-Services-Ansatz.

Das Szenario einer SOA beschreibt eine abstrakte Software-Architektur, die das Anbieten, Suchen und Nutzen von Diensten über ein Netzwerk ermöglicht. Diese Dienste lassen sich von anderen Diensten nutzen, unabhängig von der verwendeten Soft- und Hardware der einzelnen Beteiligten. Die Nutzung von einheitlichen Standards sorgt für einfaches Publizieren und Auffinden von Diensten. Die hohe Abstraktion von einer spezifischen Implementierung erleichtert eine prozessorientierte Sichtweise, da sich die Dienste funktional zerlegen lassen. Zur Integration weiterer Teilprozesse oder Dienste in die eigene Anwendung, lassen sich diese leicht über ein Netzwerk ansprechen und einbinden.
(vgl. Dostal u.a., 2004, S.7)

Dienst
Ein Dienst ist eine Softwarekomponente, die sich über ein Netzwerk erreichen und nutzen lässt. Der Zugriff auf einen Dienst wird nur über Schnittstellen ermöglicht. Diese Kapselung unterstützt das Prinzip des „information hiding", d.h. eigentliche Details der Implementierung sind für den Nutzer nicht einsehbar.

Dienstbeschreibung
Zur Nutzung des Dienstes sind die Schnittstellen öffentlich in maschinenlesbarer Form beschrieben, unabhängig von der Implementierung des Dienstes. Im Zusammenhang mit Web Services hat man hierfür WSDL (Web Service Description Language) entwickelt.

Dienstanbieter
Ein Dienstanbieter stellt seine Dienste auf einem Webserver zur Verfügung. Damit sich Dienste von potenziellen Nutzern auffinden lassen, muß der Dienstanbieter diese in einem Dienstverzeichnis registrieren. Somit ist der Dienstanbieter neben der Entwicklung einer Infrastruktur, auch für den Betrieb und die Verfügbarkeit des Dienstes zuständig.

Dienstnutzer
Der Dienstnutzer einer SOA unterscheidet sich vom traditionellen Client-Server-Ansatz. Deshalb ist wichtig, dass offene Standards vorhanden und eingehalten werden, die

Schnittstellen des Dienstes vollständig dargelegt sind und beide Seiten über gleiche Protokolle kommunizieren.

Dienstverzeichnis
Ein Dienstverzeichnis dient dem Auffinden von Diensten durch einen Dienstnutzer. Dazu muß der Dienstanbieter seine angebotenen Dienste selbstständig in einem Dienstverzeichnis registrieren, damit potenzielle Dienstnutzer diese leicht finden und nutzen können.
(vgl. Dostal u.a., 2004, S.12ff)

1.3 Web Services – Basiskomponenten SOAP, WSDL und UDDI

Die elementaren Komponenten einer SOA sind Kommunikation, Dienstbeschreibung und Dienstverzeichnis. Im Bereich von Web Services verwendet man zur Abbildung der elementaren SOA-Komponenten die folgenden Spezifikationen:

SOAP (Simple Object Access Protocol) ist die Spezifikation eines einfachen Protokolls zum Austausch strukturierter Informationen in einer Web-Service-Architektur. Ein XML-basiertes Rahmenwerk dient zur Beschreibung der Kommunikation und zur Einbettung in verschiedene Transportprotokolle. Die Verwendung von standardisierten Transportprotokollen erhöht die Interoperationalität zwischen verschiedenen Plattformen.

WSDL (Web Service Description Language) ist die Spezifikation einer XML-basierten Dienst-Beschreibungssprache, um die Schnittstellen und die angebotene Funktionalität von Web Services, die mit Hilfe von SOAP transportiert werden, zu beschreiben.

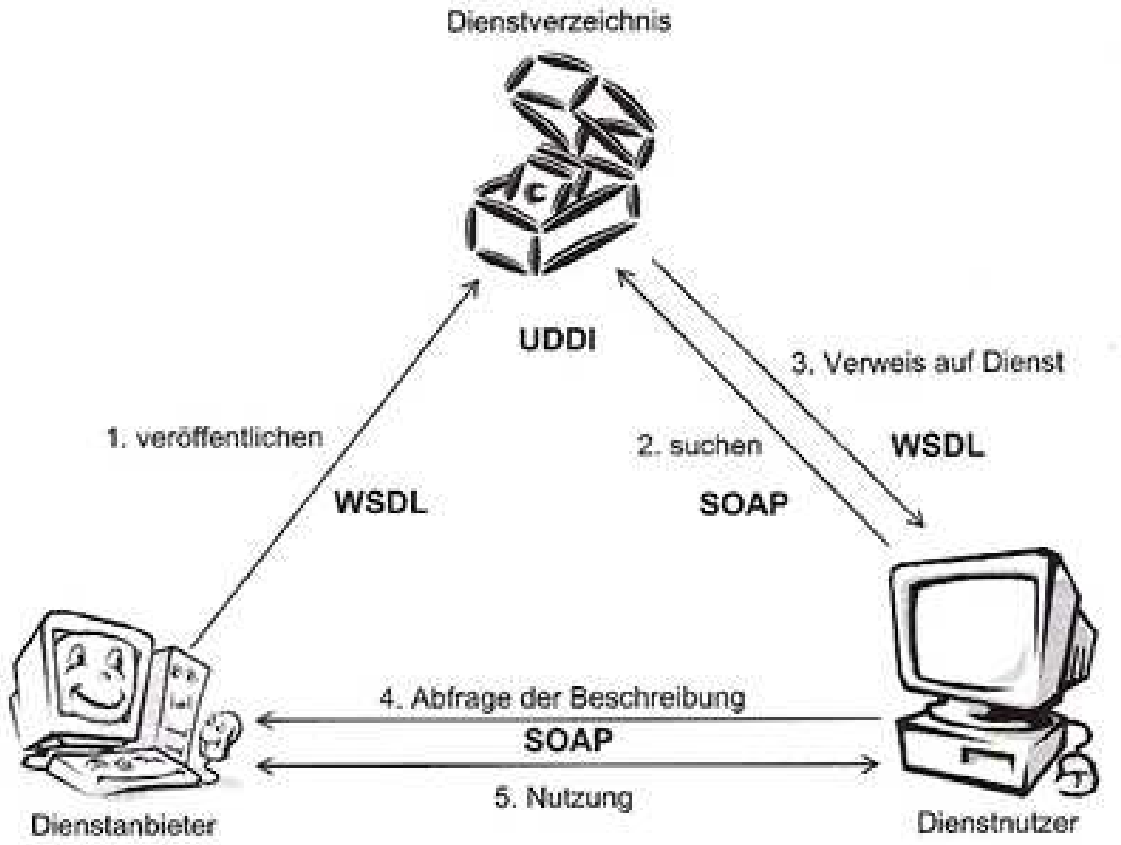
UDDI (Universal Description, Discovery and Integration) ist die Spezifikation einer standardisierten Verzeichnisstruktur für die Verwaltung von Metadaten. Diese Metadaten dienen der Beschreibung von allgemeinen Anforderungen, speziellen Dienst-Eigenschaften oder Informationen zum Auffinden von Web Services.
(Chappell & Jewell, 2002, S.3)

1.4 Web Services – Interaktionsmodell und Schichtenarchitektur

In Abbildung01 wird die Dreiecksbeziehung zwischen Dienstanbieter, Dienstverzeichnis und Dienstnutzer einer Service-orientierten Architektur, übertragen auf den Web-Service-Ansatz veranschaulicht.

Ein Dienstanbieter erstellt zum Anbieten eines Dienstes eine WSDL-Schnittstellenbeschreibung. Zur Veröffentlichung dieses WSDL-Dokuments überträgt der Dienstanbieter es an ein UDDI-basiertes Dienstverzeichnis [Schritt 1].
Wenn ein Dienstnutzer einen entsprechenden Web Service im Dienstverzeichnis sucht [Schritt 2], übermittelt das Dienstverzeichnis als Antwort das vom Dienstanbieter abgelegte WSDL-Dokument [Schritt 3].
Das übertragene WSDL-Dokument beinhaltet eine Referenz auf den entsprechenden Dienst, womit der Dienstnutzer diesen beim Dienstanbieter anfordert [Schritt 4].
Anschließend versetzt der Dienstanbieter den Dienstnutzer in die Lage, mit dem angeforderten Web Service zu kommunizieren [Schritt 5].
(vgl. Dostal u.a., 2004, S.28)

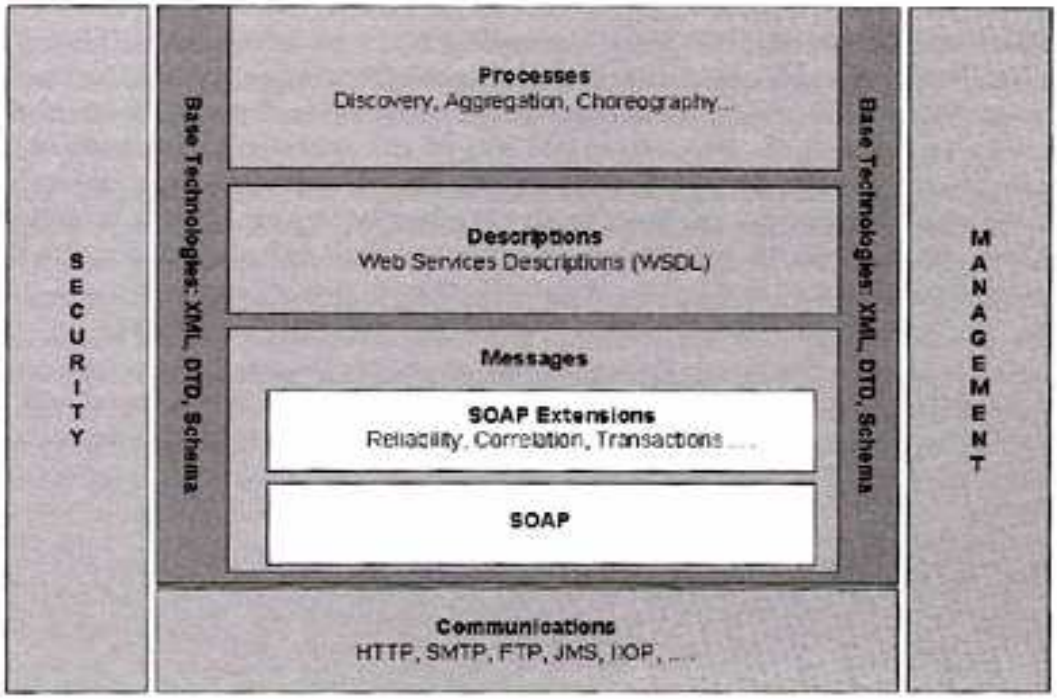
Abbildung01: Web-Services-Dreieckbeziehung *(Dostal u.a., 2004, S.28)*



Die Abbildung02 zeigt die W3C-Schichtenarchitektur von Web Services. Die Organisation W3C[2] (World-Wide-Consortium) kümmert sich im Wesentlichen um die Standardisierung von Web-Technologien. Das W3C besteht aus einer Reihe von Arbeitsgruppen, die sich jeweils mit der Standardisierung einzelner Technologien wie HTML, XML (Extensible Markup Language), DOM (Document Object Model) oder Web Services beschäftigen.
(vgl. Eberhart & Fischer, 2003, S.67)

Die Web-Services-Schichtenarchitektur basiert auf den Block „Communications“, dieser beschreibt den reinen Nachrichtenaustausch zwischen Prozessen. Als Voraussetzung gilt die Nutzung des Netzwerkprotokolls TCP/IP, möglich ist aber auch die Kommunikation über Netzwerkprotokolle wie HTTP, SMTP oder FTP.
Die zentralen Komponenten der Web-Services-Schichtenarchitektur finden sich im mittleren Block der Abbildung, eingefasst von XML, die zentrale Beschreibungssprache für Daten im Internet. Der Block „Messages“ bildet mit SOAP und einer Reihe weiterer Protokollstrukturen das Herzstück der Schichtenarchitektur. Die Schicht „Descriptions“ und die Beschreibungssprache WSDL dienen der formalen Beschreibung Web-Services-Schnittstellen. Die Schicht „Processes“ beschäftigt sich mit Definition von Prozessen, hier findet sich UDDI, ein global verfügbares Dienstverzeichnis.
Der übergreifende Block „Security“ beschäftigt sich mit der Sicherheit für Bereitstellung und Nutzung von Web Services. Die grundlegende Spezifikation WS-Security beschreibt ein Sicherheitsmodell auf Basis von Nachrichten. Der übergreifende Block „Management“ beschreibt Aspekte der Ressourcen-Verwaltung. Ein wichtiger Unterbegriff ist Dienstgüte (Quality of Service), die Einhaltung von quantitativer und qualitativer Eigenschaften.
(vgl. Eberhart & Fischer, 2003, S.71ff)

Abbildung02: Web-Services-Schichtenarchitektur *(Eberhart & Fischer, 2003, S.71)*



[2] http://www.w3.org

2.1 WSDL – Einleitung

> *„WSDL has created separate definitions and terminology for defining a web service, the communication endpoint where the web service exists, the legal format for input and output messages for the web service, and an abstract way to declare a binding to a concrete protocol and data format."*
> *(Chappell & Jewell, 2002, S.73)*

Zur Schnittstellen-Beschreibung von Web Services verwendet man WSDL (Web Services Description Language), ein standardisiertes Vokabular auf Grundlage der Beschreibungssprache XML.
WSDL beschreibt Web Services sowohl aus einer abstrakten, als auch aus einer konkreten Perspektive. Die Funktionalität von Web Services wird auf abstrakter Ebene beschrieben, technische Details auf konkreter Ebene. Die Beschreibung der Kommunikation definiert die Form von Nachrichten, die ein Web Service sendet und empfängt. Dadurch ist die Beschreibung von Web Services vollkommen unabhängig vom eventuell verwendeten Transportprotokoll oder dem Nachrichtenformat, mit dem Web Services über ein Netzwerk transportiert werden.
(vgl. Dostal u.a., 2004, S.78)

2.2 WSDL – Aufbau und Struktur

definitions
Das Wurzelelement eines WSDL-Dokuments ist das `definitions`-Element. Es ist verbindlich und darf nur einmal verwendet werden. Innerhalb des Elements finden sich optional der Name des Service als `name`-Attribut und Namespaces die angeben, welche Standards verwendet werden und helfen sollen, Elemente und ihre Verwendung zu unterscheiden.

Die wichtigsten Attribute und Namespaces:

- Das `name`-Attribut gibt den Namen des Dienstes wieder.
- Das `targetNamespace`-Attribut erlaubt einen Namespace für das gesamte WSDL-Dokument anzugeben, die Angabe ist nur als absolute URL erlaubt.
- Das `xmlns:tns`-Attribut besitzt die gleiche URL wie das `targetNamespace`-Attribut, um Namenskonflikte beim Mischen mehrerer WSDL-Dokumente zu vermeiden.
- Das `xmlns:soap`-Attribut ist der Standard-Namespace für SOAP-spezifische Informationen.
- Das `xmlns:s`-Attribut bezeichnet die Schema-Spezifikation der verwendeten Datentypen innerhalb des WSDL-Dokuments.
- Das `xmlns:http`-Attribut regelt das HTTP-Binding des WSDL-Dokuments.
- Das `xmlnx:mime`-Attribut legt einen Namespace zum Einbinden von MIME fest.
- Das `xmlns:soapenc`-Attribut bezeichnet einen Namespace des SOAP-Encoding.
- Das `xmlns`-Attribut steht für den Standard-WSDL-Namespace.

types
Für Datentypen die außerhalb der XML-Schema-Datentypen liegen, kann man im `types`-Element eigene Datentypen definieren. Dazu folgt innerhalb des `types`-Elements ein `schema`-Element, das wiederum andere Elemente zur Definition eines Datentyps enthalten kann. Die Nutzung eigener Datentypen schadet im Allgemeinen der Interoperabilität, deshalb sollte man auf die Verwendung von XML-Schema-Datentypen setzen, da diese im Standard verankert ist und sich in der Praxis optimal bewährt haben.

message
Innerhalb eines WSDL-Dokuments muss mindestens ein `message`-Element vorkommen. Ein `message`-Element besteht aus einen oder mehreren `part`-Elementen, denen man mit `type`-Attributen entsprechende Datentypen zuordnet.

portType
Das `portType`-Element enthält ein oder mehrere `operation`-Elemente. Zu jeder Methode des Web Service gehört genau ein `operation`-Element. Bei einem RPC (Remote Procedure Call), d.h. der aufrufende Client wartet bei der Ausführung einer Operation auf die Antwort des Servers, enthält neben dem `operation`-Element noch zusätzlich je ein `input`- und `output`-Element. Es gibt für die Nachrichten auch Syntax, die nicht auf RPC setzen, sie enthalten für eine einfache Nachricht nur `input`- ohne `output`-Element.

binding
Die im `portType`-Element definierten Methoden werden im `binding`-Element um die verwendeten Protokolle erweitert. Dabei können ein oder mehrere Protokolle wie SOAP, HTTP GET und HTTP POST mit einem `portType`-Element verwendet werden.
Das `binding`-Element ist wie das `portType`-Element aufgebaut, entsprechend lässt sich für jede einzelne Nachricht eines `operation`-Elements, die Bindung an Protokolle festlegen.

service
Das `service`-Element besteht aus einem oder mehreren `port`-Elementen, die Endpunkte für eine SOAP-Bindung definieren. Das `service`-Element enthält die Adresse des Dienstes für das `binding`-Element, die Verbindung erfolgt über das `binding`-Attribut des `port`-Elements.

import
Das `import`-Element ist optional und folgt direkt nach dem `definitions`-Element, es muss allerdings nicht vorhanden sein. Damit lassen sich Teile eines WSDL-Dokuments oder ganze WSDL-Dokumente importieren, dadurch lässt sich ein WSDL-Dokument flexibel aus mehreren Teilen zusammensetzen.

documentation
Das ebenfalls optionale `documentation`-Element enthält Informationen, die von Anwendungen selbst nicht angerührt werden. Sie dienen als Zusatzinformation für Programmierer, die das WSDL-Dokument lesen. Das `documentation`-Element kann innerhalb jedes anderen WSDL-Elements vorkommen.
(vgl. Hauser & Löwer, 2004, S.77ff)

2.3 WSDL – Beispiel für XML-Struktur

In Abbildung03 veranschauliche ich die elementare XML-Struktur einer WSDL-Dienstbeschreibung. Das Beispiel habe ich zufällig aus der später verwendeten WSDL-Datenmenge ausgewählt. Das WSDL-Dokument 66.wsdl beschreibt den Webdienst PacBellMessagingService[3], mit dem sich SMS-Textnachrichten verschicken lassen.

Abbildung03: 66.wsdl als Beispiel für XML-Struktur

```
<?xml version="1.0"?>
<definitions
  name="PacBellMessagingService"
  targetNamespace="http://www.xmethods.net/sd/PacBellMessagingService.wsdl"
  xmlns:tns="http://www.xmethods.net/sd/PacBellMessagingService.wsdl"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <message name="sendMessageRequest">
       <part name="From" type="xsd:string"/>
       <part name="To" type="xsd:string"/>
       <part name="Text" type="xsd:string"/>
  </message>
  <message name="sendMessageResponse">
       <part name="return" type="xsd:string"/>
  </message>
  <portType name="PacBellMessagingPortType">
       <operation name="sendMessage">
             <input message="tns:sendMessageRequest" name="sendMessage"/>
       <output message="tns:sendMessageResponse"
          name="sendMessageResponse"/>
       </operation>
  </portType>
  <binding name="PacBellMessagingBinding"
    type="tns:PacBellMessagingPortType">
    <soap:binding style="rpc"
  transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="sendMessage">
       <soap:operation soapAction="urn:xmethodsPacBellSMS#sendMessage"/>
         <input>
              <soap:body use="encoded"
                namespace="urn:xmethodsPacBellSMS"
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </input>
         <output>
              <soap:body use="encoded" namespace="urn:xmethodsPacBellSMS"
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </output>
    </operation>
   </binding>
   <service name="PacBellMessagingService">
       <documentation>
         Sends text messages to Pacific Bell SMS-capable cell phones
       </documentation>
       <port name="PacBellMessagingPort"
         binding="tns:PacBellMessagingBinding">
         <soap:address
          location="http://services.xmethods.net:80/perl/soaplite.cgi"/>
       </port>
  </service>
</definitions>
```

[3] http://www.xmethods.net/sd/PacBellMessagingService.wsdl

Ein XML-Dokument besteht aus zwei Teilen: Der Kopf liefert einem XML-Parser Informationen über die Art des Dokuments, im Rumpf befinden sich die eigentlichen Daten.

Der Kopf enthält die XML-Deklaration und besteht in unserem Beispiel aus der folgenden Zeile: `<?xml version="1.0"?>`. Die XML-Deklaration definiert die verwendete Version der XML-Spezifikation.

Der Rumpf beinhaltet den Datenbereich des XML-Dokuments. Daten repräsentiert man mit Elementen (Tags), sie treten immer paarweise auf: z.B. `<documentation>Sends text messages to […] cell phones</documentation>`.
Das eigentliche Datenelement wird vom `documentation`-Elementpaar eingeschlossen. Elemente beginnen mit „<“ und enden mit „>“, dem öffnenden und dem schließenden Element. Das schließende Element erkennt man an dem vorangestellten Schrägstrich.

Ein Element kann zusätzlich mit beliebig vielen Attributen versehen werden: z.B. `<message name="sendMessageRequest">`. Jedes Attribut besteht aus einem Schlüssel (`name`) und einem Wert (`sendMessageRequest`) in Anführungszeichen bzw. einfache Anführungszeichen gesetzt.

Es gibt zwei wichtige Regeln, die man bei der Anordnung von Elementen beachtet:

- Elemente treten immer paarweise auf, dem öffnenden Element folgt immer das korrespondierende schließende Element.
- Elementpaare dürfen sich nicht überlappen.

Das erste Element nach der XML-Deklaration im Rumpf bezeichnet man als Wurzelelement, weil es alle anderen Elemente umschließt. In WSDL-Dokumenten ist `definitions` immer das Wurzelelement.
Der große Vorteil von XML liegt in der einfachen Erweiterbarkeit. Zur Daten-Abbildung lassen sich beliebig viele Elemente definieren, was jedoch auch zu Problemen führen kann: Will man ein vorhandenes Format erweitern oder zwei XML-Dokumente zusammenführen, kann es bei der Namensgebung zu Konflikten führen. Um diesen Konflikt zu vermeiden, benutzt man Namespaces, um Elemente mit Kontext versehen.

```
xmlns:tns="http://www.xmethods.net/sd/PacBellMessagingService.wsdl"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
```

In der Zeile `xmlns="http://schemas.xmlsoap.org/wsdl/"` wird für das gesamte XML-Dokument ein Namespace definiert.
In der Zeile `xmlns:xsd="http://www.w3.org/1999/XMLSchema"` wird ein Namespace mit der Bezeichnung „`xsd`“ definiert.
(vgl. Kuschke & Wölfel, 2002 S.15ff)

2.4 WSDL – Datenquelle

Als Quelle für klassifizierte WSDL-Dokumente bietet sich die Datenquelle des ASSAM-WSDL-Annotator-Projekts[4] unter Leitung von Andreas Hess der Universität Dublin an. Ziel dieses Projekts ist die Entwicklung einer Applikation, die eine semantische Annotation von Web Services ermöglicht.
Im Internet kann man auf den Webseiten des ASSAM-WSDL-Annotator-Projekts eine Sammlung von hierarchisch klassifizierten Web Services finden. Im Rahmen des Projekts hat man 439 WSDL-Dokumente in 66 Klassen eingeordnet, die als Zip-Ordner zum freien Download bereitstehen. Die WSDL-Dokumente stammen ursprünglich von der Webseite des Dienstverzeichnis SALCentral[5] .

Neben den 439 WSDL-Dokumenten stellt das ASSAM-WSDL-Annotator-Projekt noch 439 zugehörige TXT-Dateien mit Zusatzinformationen der klassifizierten Web Services zur Verfügung:
1. Zeile: Name des Dienstanbieters
2. Zeile: original Klassifikation von SALCentral
3. Zeile: Name des Web Service im Dienstverzeichnis
4. Zeile: URL-Adresse des ursprünglichen WSDL-Dokuments
5. Zeile: Beschreibung des Web Services im SALCentral-Dienstverzeichnis
6. Zeile: ASSAM-Klassifikation [ergänzt]
(vgl. Hess, 2006)

Die 439 klassifizierten Web Services sind vom ASSAM-WSDL-Annotator-Projekt in 66 Ordner sortiert, ein einzelner Ordner repräsentiert eine Klasse. Für eine automatische Weiterverarbeitung eignet sich diese Ordnerstruktur nicht, deshalb will ich alle WSDL-Dokumente in einem Ordner darstellen. Zuvor muss ich aber die zugehörige Klassenzuordnung, die sich aus dem Namen der 66 Ordner ergeben, als Zusatzinformation in die 439 TXT-Dateien als 6. Zeile ergänzen. Dazu habe ich mit PHP ein Skript programmiert, das als Zusatzinformation die zugehörige Klasse in die TXT-Dateien ergänzt. Anschließend habe ich die WSDL-Dokumente und TXT-Dateien aus den einzelnen Ordnern entnommen und in einem gemeinsamen Ordner zusammengefasst. Um die Dateien in diesem Ordner besser verarbeiten zu können, habe ich sie schließlich einheitlich umbenannt und nummeriert.

Abbildung04: 66.txt als Beispiel für Zusatzinformationen

```
1:    xmethods
2:    Miscellaneous
3:    PacificBellSMSMessagingBridge
4:    http://www.xmethods.net/sd/PacBellMessagingService.wsdl
5:    Sends SMS messages to Pacific Bell cellular phones. This service
      allows the service client to send messages of up to 640 characters to
      cell phones that subscribe to the SMS messaging service.
6:    communication sms
```

[4] http://www.few.vu.nl/~andreas/projects/annotator
[5] http://www.servicesweb.org/servicesweb.en.php3?id_syndic=41

2.5 WSDL – Daten sammeln mit DOM-Parser

Die 439 klassifizierten WSDL-Dokumente und zugehörige TXT-Dateien sind Grundlage für die Erstellung einer Datenbank, um spezifische Eigenschaften dieser Dokumente zu entdecken und vergleichen zu können. Zur Auswahl spezifischer Eigenschaften beziehe ich mich auf die allgemeine Struktur von WSDL-Dokumenten: Ich prüfe ob die wesentlichen WSDL-Elemente `definitions`, `types`, `message`, `portType`, `binding`, `service` und `documentation` vorhanden sind und speichere die Daten, die sie repräsentieren, voneinander getrennt in einer Tabelle ab. In diese Tabelle füge ich zusätzlich die Zusatzinformationen der TXT-Dateien ein, um weitere Eigenschaften der WSDL-Dokumente zu analysieren.
Zunächst begann ich die WSDL-Datenquelle manuell zu durchsuchen, jedoch stelle ich fest, dass die manuelle Untersuchung der WSDL-Datenquelle ein mühsames und zeitraubendes Verfahren ist. Deshalb programmiere ich das PHP-Skript `DOMparser.php`, um die Datenquellen automatisch nach dem gleichen Schema zu analysieren und die gefundenen Daten in einer gemeinsamen Tabelle abzuspeichern.
Die Tabelle speichere ich im Format CSV (Character Separated Values) ab. Die Inhalte einzelner Tabellenzellen, die ermittelten WSDL-Daten, sind in einfache Anführungszeichen gesetzt und durch ein Semikolon als Trennzeichen voneinander getrennt. Um das Ende einer einzelnen Tabellenzeile anzudeuten, benutzt man einen Zeilenumbruch, um in der nächsten Zeile eine neue Tabellenzeile zu erstellen.

DOM-Modell
Für die Verarbeitung von XML-Dokumenten gibt es zwei unterschiedliche Parsertypen: Simple APi for XML (SAX) und Document Object Model (DOM).
SAX-Parser werten das XML-Dokument sequentiell aus, d.h. in der Reihenfolge ihres Auftretens werden bestimmte Elemente ausgewertet. SAX-Parser bieten nur eingeschränkte Möglichkeiten an, um die Struktur von XML Dokumenten auszulesen - die Navigation über Kind-, Vater- und Nachfolgeelemente wird nicht zur Verfügung gestellt. Der Vorteil von SAX-Parsern liegt im geringen Ressourcenverbrauch: da sich im Hauptspeicher immer nur das aktuell eingelesene Element befindet, ohne Berücksichtigung vor- oder nachgelagerter Elemente, können auch sehr große Dateien durchlesen werden. Jedoch ist zu bedenken, dass beim Lesen von Daten im hinteren Teil von sehr großen XML-Dokumenten, zuerst die vorhergehenden Elemente durchsucht werden müssen.
Bei DOM-Parsern wird das zu untersuchende XML-Dokument vor der Verarbeitung komplett in den Hauptspeicher eingelesen. Damit bieten DOM-Parser den Vorteil, innerhalb eines XML-Dokuments zu navigieren und mit Hilfe von leistungsfähigen Methoden, bestimmte Elemente zu selektieren.
(vgl. Mertens, 2003, S.225 ff)

Abbildung05: `DOM-Modell` *(Mertens, 2003, S.229)*

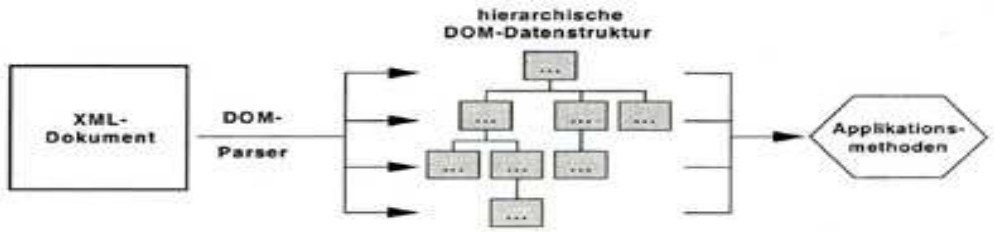


Abbildung06 zeigt die Ausgabe des PHP-Skripts DOMparser.php für den 66. Datensatz der Datenquelle. Das WSDL-Dokument wurde über die DOM-Schnittstelle eingelesen, zusätzliche ermittelte das PHP-Skript die Zusatzinformationen aus der zugehörigen TXT-Datei. Als Ausgabe erzeugte das PHP-Skript eine CSV-Datei mit den ermittelten Daten, zur Übersicht zusätzlich als Tabelle in Abbildung07 dargestellt:

Abbildung06: DOMparser-Ausgabe 66.csv

```
ID;class;provider;dict_name;wsdl_location;dict_comment;documentation;def_na
me;def_target;s_name;s_type;xsd_name;xsd_type;message_name;part_name;part_e
lement;porttype_name;operation_name;binding_name;binding_type;service_name;
soap_location;http_location;
66;'communication sms';'xmethods';'PacificBellSMSMessaging
Bridge';'http//www.xmethods.net/sd/PacBellMessagingService.wsdl';'Sends SMS
messages to Pacific Bell cellular phones. This service allows the service
client to send messages of up to 640 characters to cell phones that
subscribe to the SMS messaging service.';'Sends text messages to Pacific
Bell SMS-capable cellphones';'PacBellMessagingService';
'http//www.xmethods.net/sd/PacBellMessagingService.wsdl';'';'';'';'';'sendM
essageRequest sendMessageResponse';'From To Text return';'';
'PacBellMessagingPortType';'sendMessage sendMessage';'PacBellMessaging
Binding';'tnsPacBellMessagingPortType';'PacBellMessagingService';'http//ser
vices.xmethods.net80/perl/soaplite.cgi';'';
```

Abbildung07: Tabelle 66.csv

Eigenschaft	Wert
ID	66
Class	'communication sms'
Provider	'xmethods'
dict_name	PacificBellSMSMessagingBridge'
wsdl_location	'http//www.xmethods.net/sd/PacBellMessagingService.wsdl'
dict_comment	'Sends SMS messages to Pacific Bell cellular phones. This service allows the service client to send messages of up to 640 characters to cell phones that subscribe to the SMS messaging service.'
Documentation	'Sends text messages to Pacific Bell SMS-capable cellphones
def_name	'PacBellMessagingService'
def_target	'http//www.xmethods.net/sd/PacBellMessagingService.wsdl'
s_name	''
S_type	''
xsd_name	''
xsd_type	''
message_name	'sendMessageRequest sendMessageResponse'
part_name	'From To Text return'
part_element	''
porttype_name	'PacBellMessagingPortType'
operation_name	'sendMessage sendMessage'
binding_name	PacBellMessagingBinding'
binding_type	'tnsPacBellMessagingPortType'
service_name	'PacBellMessagingService'
soap_location	'http//services.xmethods.net80/perl/soaplite.cgi'
http_location	''

3.1 Data Mining – ökonomische Bedeutung

> *„Data mining is defined as the process of discovering patterns in data. The process must be automatic or (more usually) semiautomatic. The patterns discovered must be meaningful in that they lead to some advantage, usually an economic advantage. The data is invariably present in substantial quantities."*
> *(Witten & Frank, 2005EN, S.5)*

Data Mining beschäftigt sich mit der Entdeckung von aussagekräftigen Mustern und Zusammenhängen in Daten. Zur Auswertung von Datenmengen gibt es verschiedene Verfahren und Methoden, dazu müssen die Daten ein gewisses Maß an Struktur aufweisen, was in der Regel eine vorbereitende Aufbereitung der Daten erfordert.

Text-Klassifikation ist eine Sonderform des Data Mining und beschäftigt sich mit der Zuordnung von Dokumenten in kennzeichnende Klassen oder Kategorien. Die Daten liegen in strukturierter Form (z.B. als Tabelle) vor, deshalb lassen sich auch klassische Data-Mining-Methoden anwenden.
(vgl. Weiss u.a., 2005, S.1ff)

In der heutigen kundenbezogenen und dienstleistungsorientierten Wirtschaft ist die Auswertung von Daten ein wesentlicher Bestandteil für die Wettbewerbsfähigkeit von Unternehmen. Nach Schätzungen verdoppelt sich alle 20 Monate, die Menge der weltweit abgelegten Daten in Datenbanken. Die Welt wird immer komplexer, deshalb ist die sinnvolle Analyse dieser Daten eine wertvolle Ressource für die Wettbewerbsfähigkeit von Unternehmen.

Eine beispielhafte Anwendung des Data Mining nach, ist eine Antwort auf die folgende Frage zu bekommen: *Wie lässt sich die Kundentreue in einem stark umkämpften Marktsegment sichern?*
Der Schlüssel zur Problemlösung ist häufig eine Datenbank mit Aufzeichnungen über das Kundenverhalten und verschiedene Kundenprofile. Die Analyse der Verhaltensmuster von vergangenen Kunden lässt eindeutige Eigenschaften erkennen, welche Kunden das Produkt gewechselt und welche Kunden einem Produkt treu geblieben sind. Mit Hilfe dieser Eigenschaften versucht man, aktuelle Kunden zu identifizieren, die wahrscheinlich ebenfalls das Produkt wechseln wollen. Diese Kunden kann man mit gezielten Marketing-Methoden ansprechen und versuchen, diese davon abzuhalten.
(vgl. Witten & Frank, 2001DE, S.5)

Übertragen auf die Zielsetzung dieser Ausarbeitung stellt sich die Frage: *Lassen sich in einer Menge von WSDL-Dokumenten bestimmte Muster und Zusammenhänge erkennen, die man zur Klassifizierung von unbekannten WSDL-Dokumenten nutzen kann?*
Die bereits gesammelten Daten einer Menge von WSDL-Dokumenten untersuche ich auf potenziell kennzeichnende Eigenschaften und Muster, um die WSDL-Dokumente in aussagekräftige Klassen einzuordnen. Das Ergebnis dieser Analyse kann dann verwendet werden, um unbekannte WSDL-Dokumente auf kennzeichnende Muster und Zusammenhänge zu analysieren, um diese ebenfalls kennzeichnenden Klassen zuordnen zu können.

3.2 Data Mining – Konzepte, Instanzen & Attribute

> *„Data must be represented in a highly organized manner. We typically use spreadsheet format. The labels of columns are designed to fit the domain and are made permanent. Following this design, data are collected by adding rows (i.e., examples), where each example is measured using the same attributes."*
> *(vgl. Weiss u.a., 2005, S.3)*

Im Bereich des Data Mining gibt es vier grundsätzlich verschiedene Konzepte des maschinellen Lernens: klassifizierendes Lernen, assoziierendes Lernen, Clustering und numerische Vorhersage. Unabhängig vom verwendeten Lernverfahren bezeichnet man das, was gelernt werden soll, als Konzept.
Das klassifizierende Lernen bezeichnet Lernverfahren, die eine Menge von klassifizierten Beispielen erhalten, aus denen es lernen soll, unbekannte Beispiele zu klassifizieren. Das klassifizierende Lernen bezeichnet man auch als „überwachtes" Lernen, weil man dem Lernverfahren vorgegebene Klassifikationen zur Verfügung stellt.
Das assoziierende Lernen bezeichnet Lernverfahren, die alle Assoziationen zwischen den Attributen berücksichtigen, also nicht nur diejenigen Attribute, die eine bestimmte Klassifikation vorhersagen. Damit verfolgt das assoziierende Lernen das Ziel, interessante Strukturen in einer Menge von Daten zu finden. Das Clustering bezeichnet Lernverfahren, wo man Gruppen zusammengehöriger Beispiele sucht. Die numerische Vorhersage beschreibt Lernverfahren, die als Ergebnis keine diskrete Klasse, sondern eine numerische Größe vorhersagen. Damit ist die numerische Vorhersage eine Variante des klassifizierenden Lernens, mit einem numerischen Wert als Ergebnis.

Die Eingabe eines maschinellen Lernverfahrens bezeichnet man als Menge von Instanzen. die entweder klassifiziert, assoziiert oder gruppiert werden sollen. Jede Instanz ist ein Beispiel des zu erlernenden Konzepts, dass durch die Wertausprägungen verschiedener Attribute charakterisiert wird. Die Menge von Daten präsentiert als Tabelle voneinander unabhängiger Instanzen des zu erlernenden Konzepts.

Jede Instanz wird durch ihre Werte für eine vordefinierte Menge von Eigenschaften oder Attributen charakterisiert. Die Zeilen einer Tabelle sind Instanzen, die Spalten sind Attribute. Der Wert eines Attributs beschreibt ein Maß für die Größe, auf die sich das Attribut für eine bestimmte Instanz bezieht.
Numerische Attribute messen ganze oder reelle Zahlen. Nominale Attribute nehmen Werte einer vordefinierten, endlichen Wertemenge an. Die Werte von nominalen Größen bestehen aus unterschiedlichen Symbolen, die eigentlichen Werte dienen nur der Beschriftung, daher auch die Bezeichnung nominal, abgeleitet vom lateinischen Wort für Name. Daher ist es nicht sinnvoll, die Werte von nominalen Größen zu addieren oder multiplizieren – vielmehr lassen sich solche Attribute nur auf Gleichheit oder Ungleichheit testen.
Bei ordinalen Werten kann man eine Menge nach Kategorien sortieren. Die Sortierung folgt einer Reihenfolge, jedoch gibt es keine einheitliche Distanz zwischen den einzelnen Werten. Deshalb ist es nicht sinnvoll, Werte zu addieren oder zu subtrahieren, nur der Test auf Gleichheit oder Ungleichheit ist sinnvoll.
(vgl. Witten & Frank, 2001DE, S.41ff)

3.3 Data Mining – WEKA

> *„Experience shows that no single machine learning scheme is appropriate to all data mining problems. […] The Weka workbench is a collection of state-the-art machine learning algorithms and data preprocessing tools. […] It is designed so that you can quickly try out existing methods on new datasets in flexible ways. It provides extensive support for the whole process of experimental data mining, including preparing the input data, evaluating learning schemes statistically, and visualizing the input data and the result of learning. ”*
> *(Witten & Frank, 2005EN, S.365)*

WEKA ist eine in Java programmierte Open-Souce-Anwendung, die an der Universität von Neuseeland[6] entwickelt wurde und frei im Internet zum Download bereitsteht. WEKA bietet eine einheitliche Schnittstelle für verschiedene Lernalgorithmen, sowie Methoden zur Vor- und Nachbearbeitung von Daten und zur Evaluierung der Ergebnisse verschiedener Lernalgorithmen.
Dem Anwender präsentiert sich WEKA entweder von der Kommandozeile aus oder in Form einer graphischen Benutzeroberfläche. Diese unterteilt sich in drei voneinander unabhängigen Anwendungen, die aber alle gleichen Funktionalitäten (Filter, Klassifizierer) bieten. Das verbindende Element zwischen diesen Anwendungen ist das WEKA-eigene Format ARFF (Attribute Relation File Format). Im Rahmen meiner Untersuchung habe ich besonders die Anwendungen WEKA-Explorer und WEKA-KnowledgeFlow verwendet:

WEKA-Explorer erlaubt eine vorbereitende Betrachtung von Datenmengen und kommt mit verschiedenen Dateiformaten wie ARFF oder CSV zurecht. WEKA besitzt zusätzlich noch eine Schnittstelle, mit der sich Daten aus SQL-Datenbanken auslesen lassen.
Die in WEKA verwendeten Werkzeuge zur Vorbereitung von Daten nennt man „Filter“ und ist die wertvollste Ressource dieser Data-Mining-Software. Des Weiteren gibt es im WEKA-Explorer eine Vielzahl klassischen Data-Mining-Verfahren zur Klassifikation von Datenmengen.
WEKA-KnowledgeFlow erlaubt komplizierte Prozessablaufe modellieren zu lassen. Einzelne Datenquellen oder Klassifizierer lassen sich auf einer grafischen Oberfläche anordnen und in Beziehung setzen. Diese Daten lassen sich mit verschiedenen Lernverfahren untersuchen und mit Hilfe ausgewählter Evaluationsverfahren auswerten. Die Prozessablaufe können gespeichert werden.
(vgl. Szugat, 2005)

Abbildung08: WEKA-Explorer und WEKA-KnowledgeFlow *(vgl. Frank, 2005, S.7)*

[6] http://www.cs.waikato.ac.nz/ml/weka

3.4 Data Mining – Aufbereitung der Eingaben

Fehlerhaften und fehlenden Werten in Datenmengen sollte man große Beachtung schenken. Sie können aus unterschiedlichen Gründen auftreten: z.B. durch eine defekte Messumgebung bei der Sammlung von Daten oder beim Sammeln von Daten aus verschiedenen Datenmengen. Zwar gehen die meisten Lernverfahren implizit davon aus, dass ein fehlender Wert keine große Bedeutung für eine Instanz hat, jedoch kann es auch konkrete Gründe geben, warum genau dieser Attributwert nicht bekannt ist. Nur Personen, die mit den gesammelten Daten vertraut ist, können abschätzen, welche Bedeutung fehlenden Werten zukommt.
(vgl. Witten & Frank, 2001DE, S.52)

Im Rahmen des WEKA-Projekts an der Universität Neuseeland hat man das Dateiformat ARFF (Attribute-Relation File Format) entwickelt. Zur Beschreibung einer Menge von Instanzen, charakterisiert durch eine Menge von Attributen.
In `Abbildung08` veranschauliche ich die elementare Struktur einer ARFF-Datei.
Dazu habe ich das bereits eingeführte Beispiel `66.csv` mit Hilfe eines Texteditors in das ARFF-Format gebracht.

Abbildung08: ARFF-Datei 66.arff

```
% ARFF-Datei 66.arff
@RELATION 66.arff
@ATTRIBUTE ID numeric
@ATTRIBUTE provider string
@ATTRIBUTE dict_name string
@ATTRIBUTE wsdl_location string
@ATTRIBUTE dict_comment string
@ATTRIBUTE documentation string
@ATTRIBUTE def_name string
@ATTRIBUTE def_target string
@ATTRIBUTE s_name string
@ATTRIBUTE s_type string
@ATTRIBUTE xsd_name string
@ATTRIBUTE xsd_type string
@ATTRIBUTE message_name string
@ATTRIBUTE part_name string
@ATTRIBUTE part_element string
@ATTRIBUTE porttype_name string
@ATTRIBUTE operation_name string
@ATTRIBUTE binding_name string
@ATTRIBUTE binding_type string
@ATTRIBUTE service_name string
@ATTRIBUTE soap_location string
@ATTRIBUTE http_location string
```

```
@ATTRIBUTE class string
@DATA
66,'xmethods','PacificBellSMSMessaging
Bridge','http//www.xmethods.net/sd/PacBellMessagingService.wsdl','Sends SMS
messages to Pacific Bell cellular phones. This service allows the service
client to send messages of up to 640 characters to cell phones that
subscribe to the SMS messaging service.','Sends text messages to Pacific
Bell SMS-capable cellphones','PacBellMessagingService',
'http//www.xmethods.net/sd/PacBellMessagingService.wsdl',?,?,?,?,'sendMessa
geRequest sendMessageResponse','From To Text return',?,
'PacBellMessagingPortType','sendMessage sendMessage','PacBellMessaging
Binding','tnsPacBellMessagingPortType','PacBellMessagingService','http//ser
vices.xmethods.net80/perl/soaplite.cgi',? ,'communication sms'
```

ARFF-Dateien sind in zwei Abschnitte aufgebaut, Header- und Data-Informationen.
Zeilen, die mit einem Prozentzeichen (%) eingeleitet werden, repräsentieren Kommentare und können überall innerhalb der ARFF-Datei vorkommen.

Der Anfang des Header-Abschnitts enthält den Namen der Relation. Das Format lautet `@relation <relation-name>`, wobei `<relation-name>` vom Datentyp String ist. Wenn Leerzeichen innerhalb des Namens vorkommen, muß dieser in einfache Anführungszeichen gesetzt sein.
Die Deklaration der Attribute hat die Form `@attribute <attribute-name> <datatype>`. Für jede Spalte des Datenteils muss ein Attribut deklariert werden:
`<attribute-name>` muß mit einem Buchstaben beginnen, wenn Leerzeichen innerhalb der Attributbezeichnung vorkommen, muß diese in einfache Anführungszeichen gesetzt werden.
`<datatype>` repräsentiert vier mögliche Datentypen:

- `numeric`
  Numerische Attribute vom Typ real oder integer
- `<nominal-specification>`
  Nominale Werte repräsentieren mögliche Ausprägungen eines Attributs
- `string`
  String-Attribute repräsentieren Zeichenwerte. Wenn Leerzeichen innerhalb eines String-Attributs vorkommen, muß dieses in einfache Anführungszeichen gesetzt werden.
- `date [<date-format>]`
  Daten-Attribute werden durch das Format „yyyy-MM-dd HH:mm:ss“ repräsentiert

Der Data-Abschnitt beginnt mit `@data`, danach folgt pro Zeile eine Instanz, deren einzelne Werte durch Kommata getrennt sind. Die Werte müssen in der gleichen Reihenfolge, wie im Attribut-Abschnitt festgelegt, angegeben werden. Fehlende Werte werden durch ein Fragezeichen repräsentiert. Wenn Leerzeichen innerhalb eines einzelnen Werts vorkommen, muß dieser Wert in einfache Anführungszeichen gesetzt werden.
(vgl. WEKA, 2002)

3.5 Data Mining – Auswahl maschineller Lernverfahren

Im Laufe meiner Untersuchung verwende ich verschiedene maschinelle Lernverfahren, um die WSDL-Dokumente zu untersuchen und sie in kennzeichnende Klassen einzuordnen.

OneR - Klassifikationsregeln
Das OneR-Lernverfahren beschreibt eine sehr primitive Methode, um für eine Menge von Instanzen Klassifikationsregeln abzuleiten. Das Verfahren produziert auf der Basis eines einzigen Attributs einfache Klassifikationsregeln, die Zusammenhänge zwischen den Attributwerten und der kennzeichnenden Klasse herstellen. Obwohl es weniger sinnvoll ist, dieses Verfahren für die Klassifikation zu verwenden, kann es für die Festlegung einer Ausgangsleistung für den Vergleich mit anderen maschinellen Lernverfahren sehr sinnvolle Dienste leisten.
(vgl. Witten & Frank, 2001DE, K.8)

SMO – Support-Vektor-Maschinen
Als Erweiterung der linearen Klassifikation bezeichnet man die Anwendung von Support-Vektor-Maschinen. Ein Algorithmus auf Grundlage von linearen Modellen, um damit auch nicht lineare Klassengrenzen zu implementieren.
Lineare Modelle kann man problemlos für die Klassifikation von numerischen Attributen einsetzen. Für jede Klasse führt man eine Regression durch und ermittelt das Ergebnis der Instanzen, die zur Klasse gehören und das Ergebnis der Instanzen, die nicht zu der jeweiligen Klasse gehören. Allerdings haben lineare Modelle den Nachteil, dass sie nur lineare Grenzen zwischen Klassen repräsentieren können und deshalb allgemein für komplexe Klassifikationen zu primitiv sind.
Wie kann das Lernverfahren Support-Vektor-Maschinen auf Grundlage von linearen Modellen, aber auch nicht-lineare Klassengrenzen implementieren? Die Eingabe des Lernverfahrens wird mit Hilfe einer nicht-linearen Abbildung transformiert, d.h. der Instanzraum wird in einen neuen Raum transformiert. Leider stellt die Berechnungskomplexität von Support-Vektor-Maschinen dabei ein großes Problem dar.
(vgl. Witten & Frank, 2001DE, S.204ff)

Naive Bayes - Statistische Modellierung
Naive Bayes ist ein einfaches Lernverfahren mit klarer Semantik zur Nutzung und Erlernen von wahrscheinlichkeitsbasierten Informationen. Für viele praktische Data-Mining-Probleme erzeugt Naive Bayes beeindruckende Ergebnisse.
Das Naive-Bayes-Lernverfahren generiert auf Grundlage aller vorkommenden Attribute seine Klassifikationsergebnisse. Naive Bayes geht dabei „naiv“ von einer Unabhängigkeit der Attribute aus, das ist zwar unrealistisch, weil die Attribute in den meisten Datenmenge mit Sicherheit nicht gleich wichtig und voneinander unabhängig sind, jedoch führt das in der Praxis zu guten Klassifikationsergebnissen. Bemerkenswert ist die Tatsache, dass fehlende Werte kein Problem für das Naive-Bayes-Lernverfahren darstellen. Wenn in einer Trainingsinstanz ein Wert fehlt, wird dieser nicht in die Häufigkeitszähler aufgenommen – damit basieren die Wahrscheinlichkeitsverhältnisse nur auf Anzahl der tatsächlich vorkommenden Werte.
(vgl. Witten & Frank, 2001DE, S.88ff)

3.6 Data Mining – Evaluation der Ergebnisse

Wie kann man die Leistung von maschinellen Lernverfahren für ein bestimmtes Problem miteinander vergleichen, wenn nur eine begrenzte Anzahl von Daten vorhanden ist? Um sicher zu stellen, dass scheinbare Unterschiede nicht zufällig auftreten, benötigt man statistische Tests, um zu prüfen, wie glaubhaft die gemachten Ergebnisse sind.

Kreuzvalidierung

In Situationen, wo man praktische Klassifizierungsprobleme lösen will, ist es sinnvoll, die Leistung der einzelnen maschinellen Lernverfahren hinsichtlich einer Fehlerrate zu bestimmen. Wenn das Lernverfahren mit der Klassifikation einer einzelnen Instanz richtig liegt, wird das als Erfolg für das Lernverfahren gewertet, andernfalls als Fehler. Es stellt sich die Frage, ob die Fehlerrate für klassifizierte Daten einen sinnvollen Hinweis auf die Fehlerrate von unbekannten Daten geben kann? Die Antwort lautet nein, wenn die klassifizierten Daten zum Trainieren des Klassifizierers genutzt wurden, da die Leistungsvorhersage für die klassifizierten Daten einfach zu optimistisch ist.

Um die Leistung eines Lernverfahrens für unbekannte Daten vorherzusagen, ermittelt man die Fehlerrate für eine unabhängige Datenmenge, die man nicht zum Training des Lernverfahrens verwendet. Diese unabhängige Datenmenge bezeichnet man auch als Testmenge. Sowohl Trainingsmenge als auch Testmenge müssen repräsentative Beispiele für ein zugrunde liegendes Problem enthalten.

Wenn sehr viele Daten zur Verfügung stehen, ist die Unterteilung in Trainings- und Testmenge kein Problem. Ein Problem tritt auf, wenn nicht ausreichend Daten zur Verfügung stehen. Dann müssen Trainings- und Testdaten oft manuell klassifiziert werden, was zu einer Begrenzung aller Datenmengen führt. Deshalb verwendet man bei begrenzten Datenmengen einen bestimmten Prozentwert (Holdout) für die Testmenge vor, den Rest benutzt man für die Trainingsmenge. In der Praxis ist es üblich, ein Drittel der Daten für die Testmenge und die restlichen zwei Drittel für die Trainingsmenge zu benutzen. Dabei kann das Problem auftreten, dass die verwendete Stichprobe für das Training nicht repräsentativ ist. Deshalb sollte vorher überprüft werden: Ist jede Klasse aus der gesamten Datenmenge im richtigen Verhältnis der Trainings- und Testmengen? Diese Prozedur bezeichnet man als Stratifikation (Schichtenbildung), jedoch bildet sie nur einen einfachen Schutz gegen ungleiche Repräsentation in Trainings- und Testdatenmengen.

Eine bessere Methode, um die Fehlerrate einer begrenzten Datenmenge vorhersagen, ist die Anwendung der zehnfachen Kreuzvalidierung. Die Daten werden in zehn Teile zerlegt, wobei jeder Teil annähernd das Klassenverhältnis der gesamten Datenmenge repräsentiert. Anschließend verwendet man nacheinander, jeden Teil als Testmenge und den Rest für die Trainingsmenge und ermittelt die einzelnen Fehlerraten. Insgesamt ermittelt man also für zehn unterschiedliche Trainings- und Testmengen die Fehlerraten, mit deren Durchschnitt sich ein Gesamtfehler berechnen lässt.

(vgl. Witten & Frank, 2001DE, S.127ff)

4.1 Evaluation der Daten

Modellierung der ARFF-Daten

Die mit Hilfe des DOM-Parser erstellte CSV-Tabelle mit den WSDL-Daten enthält 439 Instanzen, nach Vorschlag des ASSAM-WSDL-Annotator-Projekts in 66 Klassen eingeteilt. Neben dem Klassenattribut gibt es noch 21 Attribute, die jeweilige Eigenschaften der 439 WSDL-Dokumente repräsentieren, sowie Zusatzinformationen zur Quelle der WSDL-Dokumente beinhalten.

Da sich innerhalb der CSV-Tabelle einige Instanzen befinden, die nicht ins Format passen, weil einzelne Attributwerte zu viele Zeichen enthalten, müssen diese Instanzen entfernt werden. Diese Datensäuberung ist nötig, damit WEKA die Daten verarbeiten kann. Insgesamt entferne ich von den 439 Instanzen genau 10 Instanzen, die aufgrund zu langer Attributwerte nicht in das Format der CSV-Tabelle passen. Damit befinden sich noch 429 Instanzen in der CSV-Tabelle, ausgehend von der ursprünglichen CSV-Tabelle entferne ich also 2,28 % fehlerhafter Instanzen.

Bei der Umwandlung der CSV-Tabelle in eine ARFF-Datei mit Hilfe eines Text-Editors, zeigt sich, dass das `documentation`-Attribut Probleme erzeugt. Innerhalb der `documentation`-Spalte befinden sich einfache Anführungszeichen, doppelte Anführungszeichen oder HTML-Tags, die bei einer Konvertierung in das ARFF-Format innerhalb von WEKA zu Fehlermeldungen führen. Diese Zeichen manuell aus der CSV-Tabelle zu entfernen, würde einen hohen zeitlichen Aufwand bedeuten, deshalb entscheide ich mich, das gesamte `documentation`-Attribut aus meiner Untersuchung zu entfernen. Damit befinden sich neben dem Klassenattribut noch 20 Attribute in der CSV-Tabelle. Diese CSV-Tabelle lässt sich ohne Probleme in eine ARFF-Datei umwandeln, damit WEKA sie auslesen und verarbeiten kann. Dazu benutze ich einen Text-Editor, um alle nacheinander folgenden einfachen Anführungszeichen in Fragezeichen umzuwandeln, da diese fehlende Werte innerhalb einer Instanz repräsentieren. Anschließend ersetze ich alle vorkommenden Semikolons als Trennzeichen durch Kommata und ergänze die fehlenden ARFF-Informationen, um sie als ARFF-Datei abspeichern zu können.

Die erstellte ARFF-Datei enthält 20 WSDL-Attribute und ein Klassenattribut. Da WEKA bei den verwendeten Lernverfahren nicht mit String-Attribute umgehen kann, muss ich die von WEKA bereitgestellten Filter benutzen, um die String-Attribute der ARFF-Datei in andere Maßeinheiten umzuwandeln, um sie in WEKA nutzen zu können.

Für die Umwandlung des Klassenattributs gibt es in WEKA den Filter `StringToNominal`, der alle vorkommenden String-Klassen in nominale Werte umwandelt. Nach Anwendung erhalte ich ein nominales Klassenattribut, das zwischen 66 Klassen unterscheidet.

In Tabelle `Abbildung09` sind diese 66 Klassen mit der Anzahl der zugehörigen Instanzen aufgeführt, z.B. gibt es 13 von 429 Instanzen, die Klasse `business` zugeordnet sind.

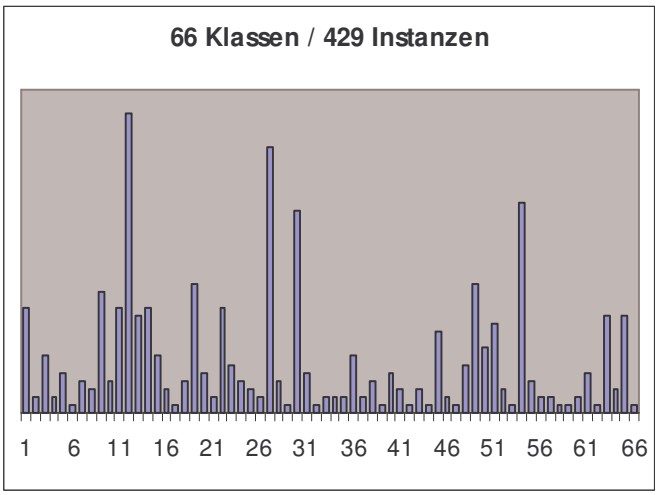
Abbildung09: `66 Klassen`

#	Klassenbezeichnung	Anz
1	business	13
2	business charts	2

#	Klassenbezeichnung	Anz
34	finder shakespeare	2
35	finder stars	2

3	business customers	7	36	finder whoIS	7
4	cartoon	2	37	Finder xml_xhtmlmaker	2
5	communication	5	38	Flights	4
6	communication calling cards	1	39	Flights airportInfo	1
7	communication fax	4	40	games	5
8	communication instantMessaging	3	41	games chess	3
9	communication mail	15	42	Genbank	1
10	communication emailvalidation	4	43	graphics	3
11	communication sms	13	44	License	1
12	converter	37	45	mathematics	10
13	converter dictionary	12	46	money	2
14	countryInfo	13	47	money auction	1
15	countryInfo cityInformation	7	48	money bank credit card	6
16	countryInfo cityInformation addressValidation	3	49	money bank stock	16
17	countryInfo cityInformation office	1	50	money business	8
18	countryInfo cityInformation phone	4	51	money currency	11
19	countryInfo cityInformation postcodes	16	52	money ePayment	3
20	countryInfo cityInformation postcodes distance	5	53	music	1
21	countryInfo routes	2	54	news	26
22	countryInfo weather	13	55	News quotes	4
23	courierInfo	6	56	Parser	2
24	databaseprovider	4	57	pollcreation	2
25	datamanagement	3	58	serverInfo	1
26	datamanagement contentmanagement	2	59	usergroups	1
27	developers	33	60	web	2
28	developers encryption	4	61	web domain	5
29	engineering	1	62	web ip	1
30	finder	25	63	web webservice	12
31	finder book	5	64	web webservice uddi	3
32	finder links	1	65	web websites	12
33	finder phrases	2	66	web websitecreation	1

Abbildung10: 66 Klassen / 429 Instanzen



Im Diagramm Abbildung10 sieht man, wie sich die 429 Instanzen der Trainingsdaten auf die 66 Klassen verteilen.
Es gibt Klassen, denen viele Instanzen zugeordnet sind (z.B. converter mit 37 Instanzen). Es gibt aber auch Klassen, denen nur wenige Instanzen zugeordnet (z.B. engineering mit 1 Instanz).

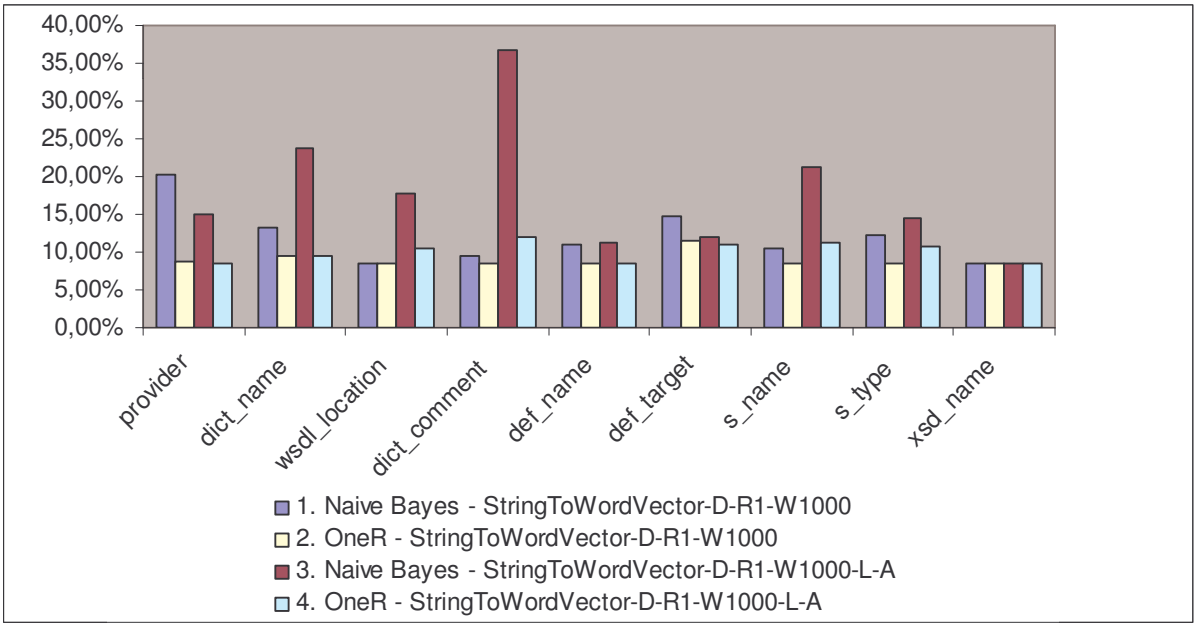
4.2 Untersuchung 1 – Umwandlung der String-Attribute

Die 20 String-Attribute zur Beschreibung der WSDL-Eigenschaften muss ich ebenfalls in andere Maßeinheiten umwandeln, damit WEKA mit diesen Attributen umgehen kann. WEKA stellt den Filter `StringToWordVector` zur Verfügung, der vorkommende String-Attribute in numerische Attribute umwandelt.

Frage: *Soll ich die String-Attribute unverändert mit allen Zeichen in numerische Attribute umwandeln oder soll ich nur die Buchstaben innerhalb der String-Attribute berücksichtigen, um sie in numerische Attribute umzuwandeln?*

Um eine Antwort auf diese Frage zu bekommen, erzeuge ich neun ARFF-Dateien aus jeweils einem String-Attribut und dem nominalen Klassenattribut bestehen. Auf Grundlage dieser neun ARFF-Dateien erstelle ich mit Hilfe der WEKA-KnowledgeFlow insgesamt 18 ARFF-Dateien, die keine String-Attribute mehr enthalten. Dabei erzeuge ich neun ARFF-Dateien, wo die String-Attribute unverändert mit allen Zeichen in numerische Werte umgewandelt sind und neun ARFF-Dateien, wo die String-Attribute nur unter Berücksichtigung von Buchstaben umgewandelt sind.

Abbildung11: `Klassifikation der String-Attribute`



Das Diagramm `Abbildung11` zeigt die Klassifikationsergebnisse der 18 ARFF-Dateien mit den Lernverfahren Naive Bayes und OneR, angewendet auf die 429 Instanzen der Trainingsdatenmenge, evaluiert mit zehnfacher Kreuzvalidierung.

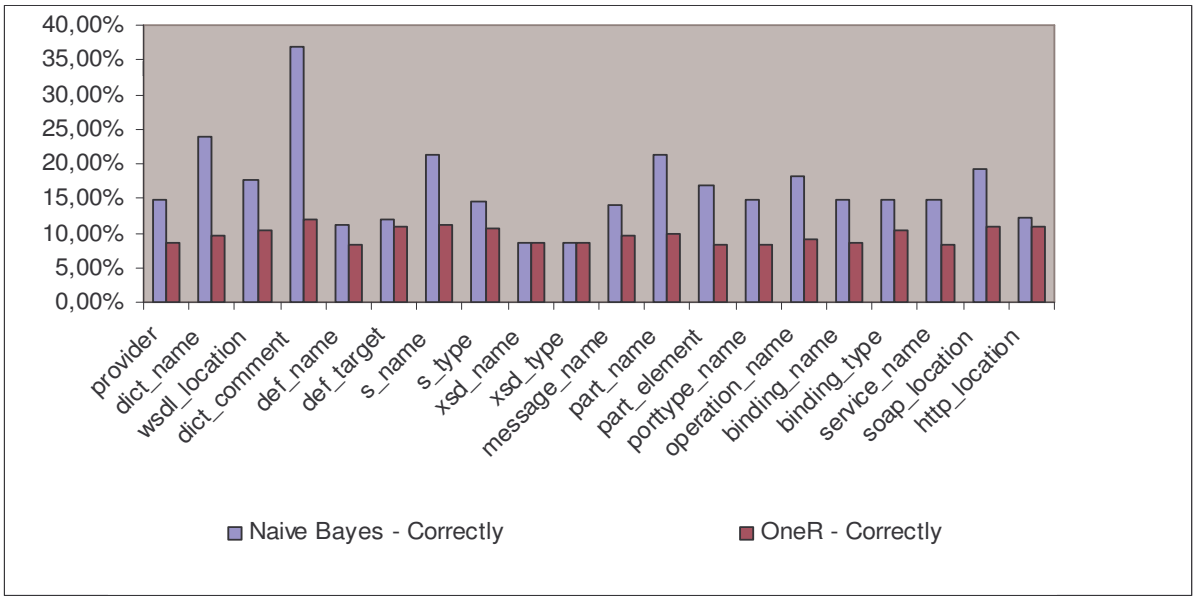
Ich stelle fest, dass die neun ARFF-Dateien, in denen nur Buchstaben für die Umwandlung der String-Attribute verwendet werden, tendenziell bessere Klassifikationsergebnisse liefern, als die neun ARFF-Dateien, in denen die String-Attribute unverändert mit allen Zeichen in numerische Werte umgewandelt wurden. Deshalb beschließe ich, für weitere Untersuchungen der WSDL-Attribute die String-Attribute nur noch unter Berücksichtigung der Buchstaben in numerische Attribute zu betrachten.

4.3 Untersuchung 2 – Evaluation der WSDL-Attribute

Frage: *Welches WSDL-Attribut liefert die besten Klassifikationsergebnisse, wenn man die einzelnen WSDL-Attribute miteinander vergleicht?*

Um eine Antwort auf Frage zu erhalten, erzeuge ich zusätzlich zu den neun ARFF-Dateien aus Untersuchung 1, für die restlichen elf WSDL-Attribute jeweils eine ARFF-Datei, bestehend aus einem einzelnen WSDL-Attribut und dem nominalen Klassenattribut. Auf Grundlage dieser elf ARFF-Dateien erstelle ich mit Hilfe der WEKA-KnowledgeFlow entsprechende ARFF-Dateien ohne String-Attribute. Dabei stütze ich mich auf Untersuchung 1 und berücksichtige für die Umwandlung der String-Attribute in numerische Attribute nur die Buchstaben innerhalb der Attributwerte. Die 20 ARFF-Dateien für die einzelnen WSDL-Attribute untersuche ich mit den Lernverfahren Naive Bayes und OneR, somit kann ich die Ergebnisse aus Untersuchung 1 teilweise wieder verwenden.

Abbildung12: Klassifikation der WSDL-Attribute



Das Diagramm Abbildung12 zeigt die Klassifikationsergebnisse für die 20 ARFF-Dateien mit den Lernverfahren Naive Bayes und OneR, angewendet auf die 429 Instanzen der Trainingsdatenmenge, evaluiert mit zehnfacher Kreuzvalidierung.

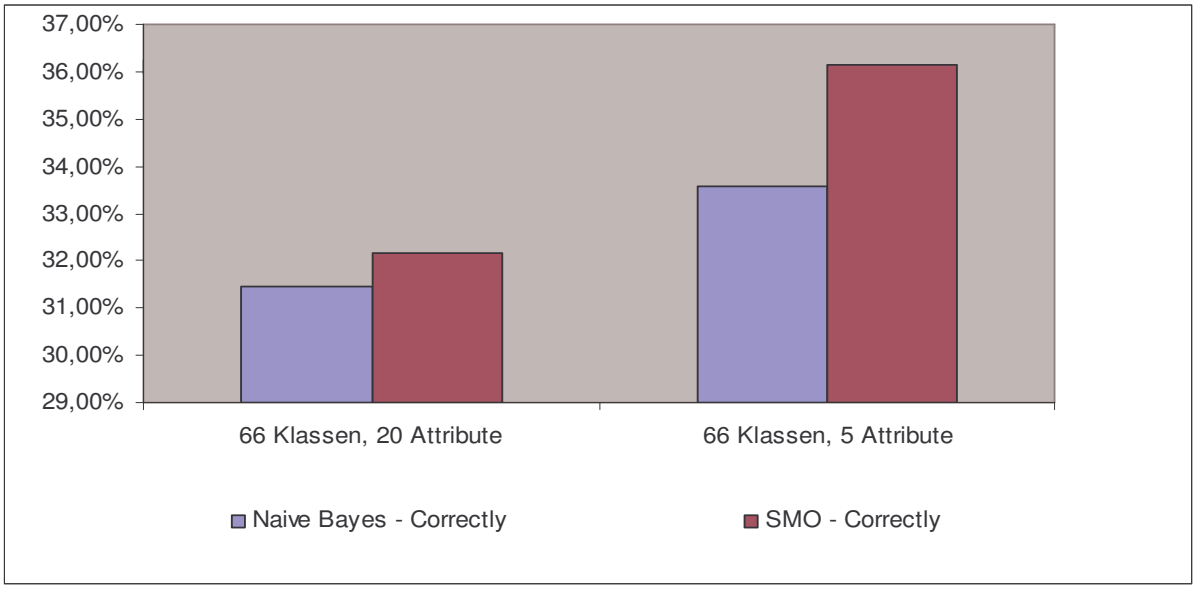
Ich stelle fest, dass sich einige WSDL-Attribute besser zur Klassifikation eignen, als andere WSDL-Attribute. Besonders die WSDL-Attribute `dict_name`, `dict_comment`, `s_name`, `part_name`, `operation_name` und `soap_location` liefern für die Klassifikation mit dem Naive-Bayes-Lernverfahren besonders gute Ergebnisse, mit min. 20% korrekt klassifizierter Instanzen. Diesen Trend stellt man ebenfalls beim OneR-Lernverfahren fest, auch wenn es schlechtere Klassifikationsergebnisse liefert.
Für weitere Untersuchungen verwende ich die WSDL-Attribute `dict_comment`, `s_name`, `part_name`, `operation_name` und `soap_location`, weil diese Attribute die reinen 429 WSDL-Dateien repräsentieren. Das WSDL-Attribut `dict_comment` ist zwar ursprünglich von den Zusatzinformationen der TXT-Dateien entnommen, repräsentiert aber am ehesten, das aus der Untersuchung entfernte `documentation`-Attribut.

4.4 Untersuchung 3 – Evaluation mit 66 Klassen

Frage: *Wie sinnvoll ist die Einteilung in 66 Klassen?*

Um eine Antwort auf die Frage zu bekommen, ermittle ich, welche Ergebnisse die Klassifikation mit der 66-Klasseneinteilung liefert, um sie mit anderen Einteilungen zu vergleichen.
Ich erzeuge eine ARFF-Datei, die aus 20 WSDL-Attributen im String-Format und dem nominalen Klassenattribut bestehen. Die String-Attribute der ARFF-Datei wandle ich mit Hilfe des WEKA-Explorer in numerische Attribute um, dabei stütze ich mich auf die Ergebnisse von Untersuchung 1 und berücksichtige nur die Buchstaben innerhalb der String-Attribute. Zusätzlich erstelle ich eine modifizierte ARFF-Datei, dabei stütze ich mich auf die Ergebnisse von Untersuchung 2 und berücksichtige nur die WSDL-Attribute `dict_comment`, `s_name`, `part_name`, `operation_name` und `soap_location`. Auch die String-Attribute dieser ARFF-Datei wandle ich mit Hilfe des WEKA-Explorer in entsprechende numerische Attribute um.

Abbildung13: Klassifikation der 66 Klassen



Das Diagramm `Abbildung13` zeigt die Klassifikationsergebnisse für die beiden ARFF-Dateien mit 20 bzw. fünf WSDL-Attributen, eingeteilt in 66 Klassen. Verwendet wurden die Lernverfahren Naive Bayes und Support-Vektor-Maschinen, angewendet auf die 429 Instanzen der Trainingsdatenmenge, evaluiert mit zehnfacher Kreuzvalidierung.

Ich stelle fest, dass die ARFF-Datei mit fünf WSDL-Attributen für beide Lernverfahren bessere Klassifikationsergebnisse liefert als die ARFF-Datei mit 20 WSDL-Attributen. Das stützt meine Ergebnisse von Untersuchung 2, dass sich bestimmte WSDL-Attribute besonders gut für die Klassifikation von WSDL-Dokumenten eignen.
Außerdem stelle ich fest, dass die Klassifikation mit den Support-Vektor-Maschinen-Verfahren bessere Ergebnisse als das Naive-Bayes-Verfahren liefert.
Insgesamt sind die Ergebnisse dieser Klassifikation nicht zufrieden stellend: ein Niveau von ca. 35% korrekt klassifizierter Instanzen bedeutet andererseits, dass ca. 65% aller Instanzen falsch klassifiziert sind.

4.5 Untersuchung 4 – Evaluation mit 11 Klassen

Frage: *Lassen sich mit weniger Klassen bessere Klassifikationsergebnisse erreichen?*

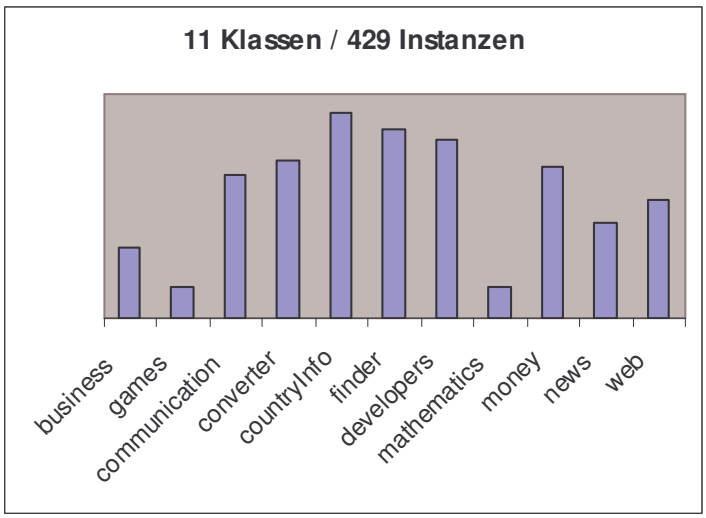
Da das Ergebnis von Untersuchung 3 mit ca. 35% korrekt klassifizierte nicht zufrieden stellend ist, stellt sich die Frage, ob man mit weniger Klassen die Klassifikationsergebnisse verbessern kann. Wenn man sich die Aufteilung der Instanzen auf die 66 Klassen am Anfang der Untersuchung veranschaulicht, stellt man fest, dass viele Klassen Unterkategorien besitzen. Deshalb will ich die Unterkategorien entfernen und sie einer gemeinsamen Kategorie zuweisen, was möglicherweise zu besseren Klassifikationsergebnissen führt.

In Tabelle `Abbildung14` sind die 66 Klassen von Untersuchung 3 in elf Klassen vereinfacht und mit der Anzahl der zugehörigen Instanzen aufgeführt, z.B. gibt es 22 von 429 Instanzen, die Klasse `business` zugeordnet sind.

Abbildung14: `11 Klassen`

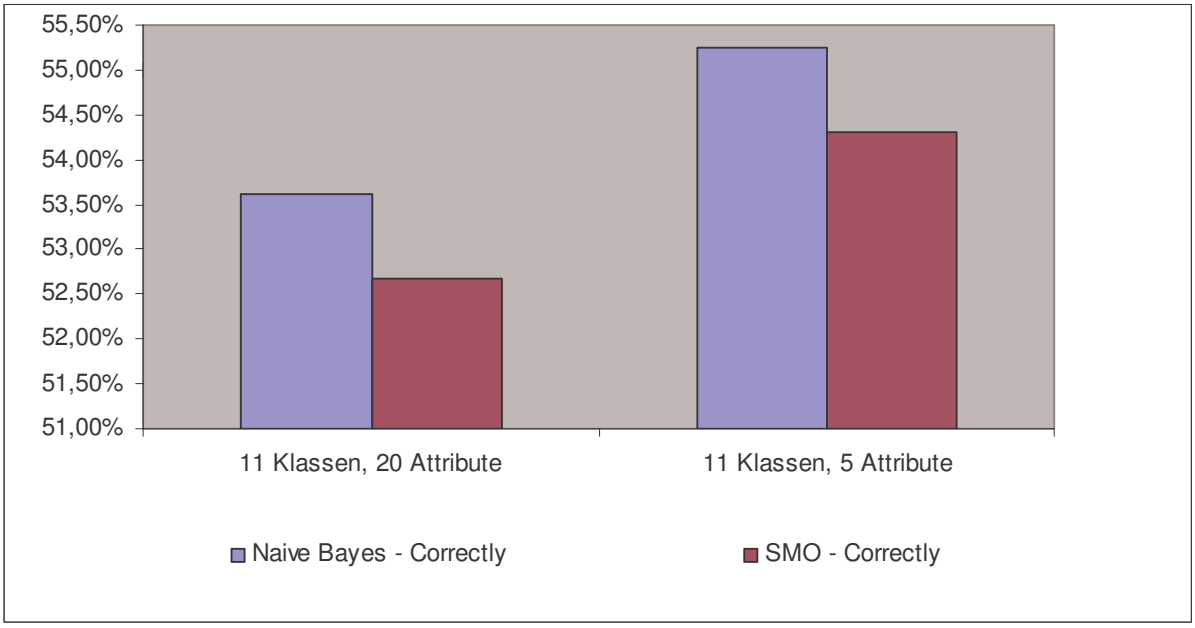
#	Klassenbezeichnung	enthält Klassen…	Anz
1	business	business & Unterkategorien (22)	22
2	games	games & Unterkategorien (8), cartoon (2)	10
3	communication	communication & Unterkategorien (45)	45
4	converter	converter & Unterkategorien (49)	49
5	countryInfo	countryInfo & Unterkategorien (64)	64
6	finder	finder & Unterkategorien (46), courierInfo (6), flights & Unterkategorien (5), genbank (1), music (1)	59
7	developers	developers & Unterkategorien (37),databaseprovider (4), datamanagement & Unterkategorien (5),engineering (1), graphics (3), license (1), parser (2), pollcreation (2), usergroups (1)	56
8	mathematics	mathematics (10)	10
9	money	money & Unterkategorien (47)	47
10	news	news & Unterkategorien (30)	30
11	web	Web & Unterkategorien (36), serverInfo (1)	37

Abbildung15: `11 Klassen / 429 Instanzen`



Im Diagramm `Abbildung15` sieht man, wie sich die 429 Instanzen der Trainingsmenge auf die elf Klassen verteilen.
Die Instanzen sind nun besser auf die Klassen ausbalanciert, jede Klasse besitzt mindestens zehn Instanzen. Den Klassen `games` und `mathematics` ist die geringste Anzahl von Instanzen zugeordnet. Der Klasse `countryInfo` sind die meisten Instanzen zugeordnet.

Abbildung16: Klassifikation der 11 Klassen



Das Diagramm Abbildung16 zeigt die Klassifikationsergebnisse für die beiden ARFF-Dateien mit 20 bzw. fünf WSDL-Attributen, eingeteilt in elf Klassen. Verwendet wurden die Lernverfahren Naive Bayes und Support-Vektor-Maschinen, angewendet auf die 429 Instanzen der Trainingsdatenmenge, evaluiert mit zehnfacher Kreuzvalidierung.

Ich stelle fest, dass die ARFF-Datei mit fünf WSDL-Attributen für beide Lernverfahren wieder einmal bessere Klassifikationsergebnisse liefert als die ARFF-Datei mit 20 WSDL-Attributen. Das stützt meine Ergebnisse von Untersuchung 2, dass sich bestimmte WSDL-Attribute besonders gut für die Klassifikation von WSDL-Dokumenten eignen.
Außerdem stelle ich fest, dass die Klassifikation mit den Naive-Bayes-Verfahren diesmal bessere Ergebnisse als das Support-Vektor-Maschinen-Verfahren liefert.
Insgesamt sind die Ergebnisse dieser Klassifikation eher zufrieden stellend, als bei Untersuchung 3: ein Niveau von ca. 54% korrekt klassifizierter Instanzen bedeutet andererseits, dass nur noch ca. 46% aller Instanzen falsch klassifiziert sind.

4.6 Untersuchung 5 – Evaluation mit 5 Klassen

Frage: *Lassen sich mit weniger Klassen nochmals bessere Klassifikationsergebnisse erreichen?*

Da das Ergebnis von Untersuchung 4 mit ca. 54% korrekt klassifizierter Instanzen so viel besser als das Ergebnis von Untersuchung 3 mit ca. 35% ist, stellt sich die Frage, ob man mit weniger Klassen die Klassifikationsergebnisse nochmals verbessern kann.
Um eine Antwort auf die Frage zu bekommen, welche Klassen sich nochmals zusammenfassen lassen, kann man eine Korrelationsmatrix zu Rate ziehen. In einer Korrelationsmatrix sieht man, welche Klassen untereinander korrelieren, d.h. es lassen sich Zusammenhänge zwischen der eigentlichen Klassenzuordnung von Instanzen und der Klassenzuteilung als Ergebnis einer Klassifikation aufzeigen. Die größten Werte innerhalb der Korrelationsmatrix weisen auf die größten Zusammenhänge von Klassen hin.

`Abbildung17` zeigt die Korrelationsmatrix für die 11-Klassen-Trainingsmenge der ARFF-Datei mit fünf WSDL-Attributen, als Ergebnis der Klassifikation mit dem Naive-Bayes-Verfahren in der 4. Untersuchung. In der Korrelationsmatrix lässt sich beispielsweise ablesen, das von allen 22 Instanzen der Klasse `business` als Ergebnis der Klassifikation, nur eine Instanz tatsächlich der Klasse `business` zugeordnet wurde. Dagegen wurden die restlichen 21 Instanzen der Klasse `business` als Ergebnis der Klassifikation, falschen Klassen zugeordnet: 3 Instanzen der Klasse `communication`, 4 Instanzen der Klasse `countryInfo`, 3 Instanzen der Klasse `finder`, 8 Instanzen der Klasse `developers`, 2 Instanzen der Klasse `money` und 1 Instanz der Klasse `web`.

Abbildung17: `Korrelationsmatrix für 11 Klassen`

```
 a  b  c  d  e  f  g  h  i  j  k   <-- classified as
 1  0  3  0  4  3  8  0  2  0  1 |  a = business
 0  2  0  0  1  2  3  0  1  1  0 |  b = games
 1  0 39  0  1  1  2  0  0  0  1 |  c = communication
 1  0  2 31  1  1  6  2  3  1  1 |  d = converter
 1  0  0  1 49  5  2  0  2  3  1 |  e = countryInfo
 0  0  0  1  9 30 11  0  5  3  0 |  f = finder
 2  0  5  4  0 13 25  2  3  1  1 |  g = developers
 0  0  0  1  0  1  3  4  0  1  0 |  h = mathematics
 0  0  2  3  1  5  4  0 30  0  2 |  i = money
 1  0  0  1  2  8  3  0  4 11  0 |  j = news
 1  0  2  1  1  8  6  0  1  2 15 |  k = web
```

Die markierten Zahlen zeigen die größten Zusammenhänge innerhalb der Korrelationsmatrix auf. Besonders zwischen den Klassen `finder` und `developers` besteht ein großer Zusammenhang, d.h. wenn man diese beiden Klassen zusammenfasst, werden weniger Instanzen falsch klassifiziert, was insgesamt zu mehr korrekt klassifizierten Instanzen führt.

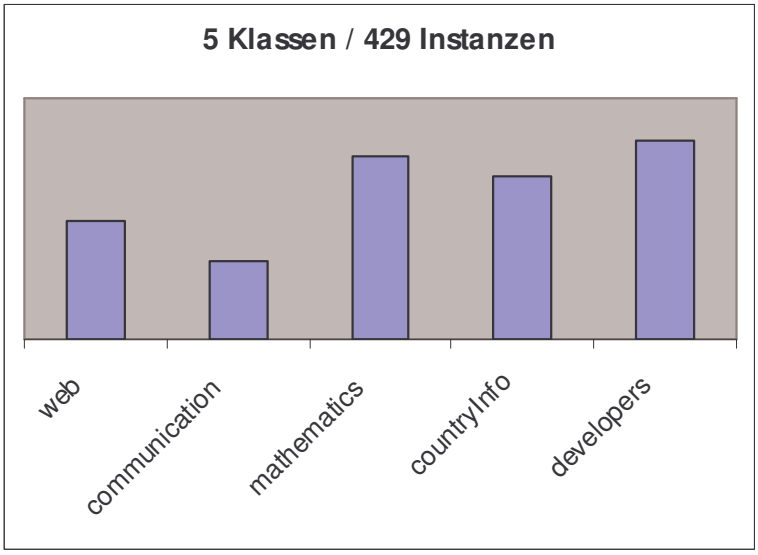
In Tabelle `Abbildung18` sind die elf Klassen von Untersuchung 4 in fünf Klassen vereinfacht und mit der Anzahl der zugehörigen Instanzen aufgeführt, z.B. gibt es 69 von 429 Instanzen, die Klasse `web` zugeordnet sind. Dabei stütze ich mich sowohl auf die Aufwertung der vorangestellten Korrelationsmatrix, als auch auf Überlegungen, ob Klassen möglicherweise inhaltlich dasselbe repräsentieren. So repräsentieren die Klassen

converter, mathematics und money inhaltlich weitgehend dasselbe, alle drei Klassen beschäftigen sich irgendwie mit Zahlen.

Abbildung18: 5 Klassen

#	Klassenbezeichnung	enthält Klassen…	Anz
1	web	web & Unterkategorien (36), serverInfo (1), business & Unterkategorien (22), games & Unterkategorien (8), cartoon (2)	69
2	communication	communication & Unterkategorien (45)	45
3	mathematics	mathematics (10), converter & Unterkategorien (49), money & Unterkategorien (47)	106
4	countryInfo	countryInfo & Unterkategorien (64), news & Unterkategorien (30)	94
5	developers	developers & Unterkategorien (37),databaseprovider (4), datamanagement & Unterkategorien (5),engineering (1), graphics (3), license (1), parser (2), pollcreation (2), usergroups (1), finder & Unterkategorien (46), courierInfo (6), flights & Unterkategorien (5), genbank (1), music (1)	115

Abbildung19: 5 Klassen / 429 Instanzen



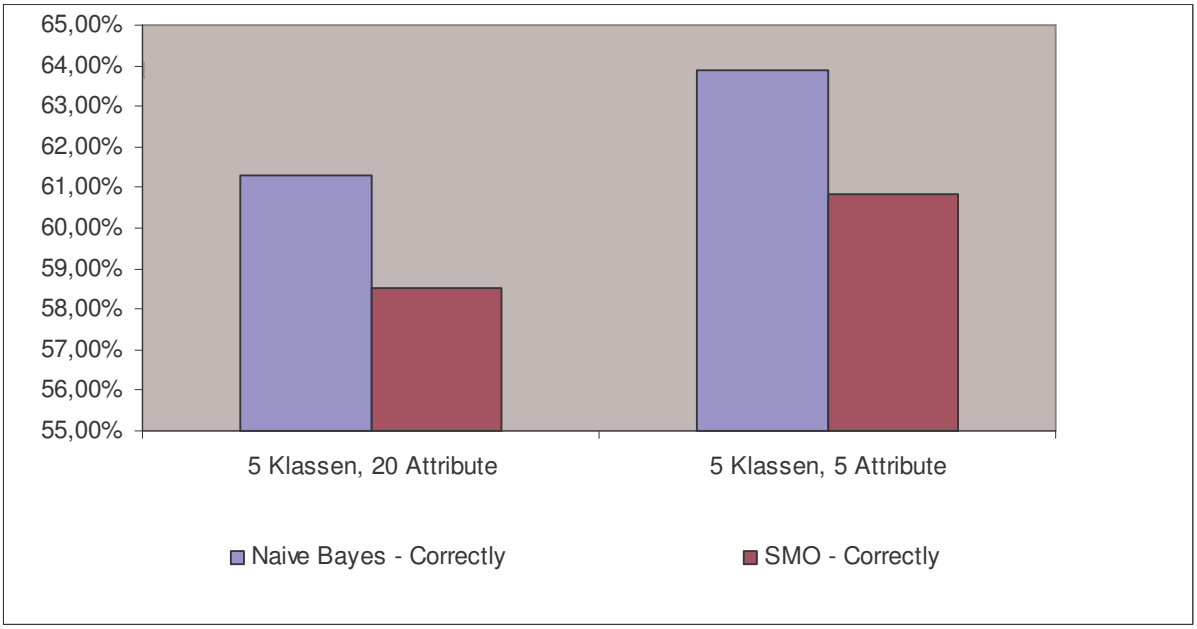
Im Diagramm Abbildung19 sieht man, wie sich die 429 Instanzen der Trainingsmenge auf die fünf Klassen verteilen.
Die Instanzen sind nochmals besser ausbalanciert, verteilen sich nun gleichmäßiger auf die fünf Klassen.

Das Diagramm Abbildung20 zeigt die Klassifikationsergebnisse für die beiden ARFF-Dateien mit 20 bzw. fünf WSDL-Attributen, eingeteilt in fünf Klassen. Verwendet wurden die Lernverfahren Naive Bayes und Support-Vektor-Maschinen, angewendet auf die 429 Instanzen der Trainingsdatenmenge, evaluiert mit zehnfacher Kreuzvalidierung.

Ich stelle fest, dass die ARFF-Datei mit fünf WSDL-Attributen für beide Lernverfahren wieder einmal bessere Klassifikationsergebnisse liefert als die ARFF-Datei mit 20 WSDL-Attributen. Das stützt meine Ergebnisse von Untersuchung 2, dass sich bestimmte WSDL-Attribute besonders gut für die Klassifikation von WSDL-Dokumenten eignen.
Außerdem stelle ich fest, dass die Klassifikation mit den Naive-Bayes-Verfahren auch diesmal bessere Ergebnisse als das Support-Vektor-Maschinen-Verfahren liefert.

Insgesamt sind die Ergebnisse dieser Klassifikation nochmals besser als bei Untersuchung 4: ein Niveau von ca. 62% korrekt klassifizierter Instanzen bedeutet andererseits, dass nur noch ca. 38% aller Instanzen falsch klassifiziert sind.

Abbildung20: Klassifikation der 5 Klassen



4.7 Untersuchung 6 – Evaluation mit 4 Klassen

Frage: *Lassen sich mit weniger Klassen nochmals bessere Klassifikationsergebnisse erreichen?*

Da das Ergebnis von Untersuchung 5 mit ca. 62% korrekt klassifizierter Instanzen so viel besser als das Ergebnis von Untersuchung 3 mit ca. 35% ist, stellt sich die Frage, ob man mit weniger Klassen, die Klassifikationsergebnisse nochmals verbessern kann.

Abbildung21 zeigt die Korrelationsmatrix für die 5-Klassen-Trainingsmenge der modifizierten ARFF-Datei mit fünf Attributen, als Ergebnis der Klassifikation mit dem Naive-Bayes-Verfahren in der 5. Untersuchung.

Abbildung21: Korrelationsmatrix für 5 Klassen

```
 a  b  c  d  e   <-- classified as
27  6  6  8 22 |  a = web
 7 34  1  0  3 |  b = communication
 8  3 76  4 15 |  c = mathematics
 5  0  3 70 16 |  d = countryInfo
17  7 11 13 67 |  e = developers
```

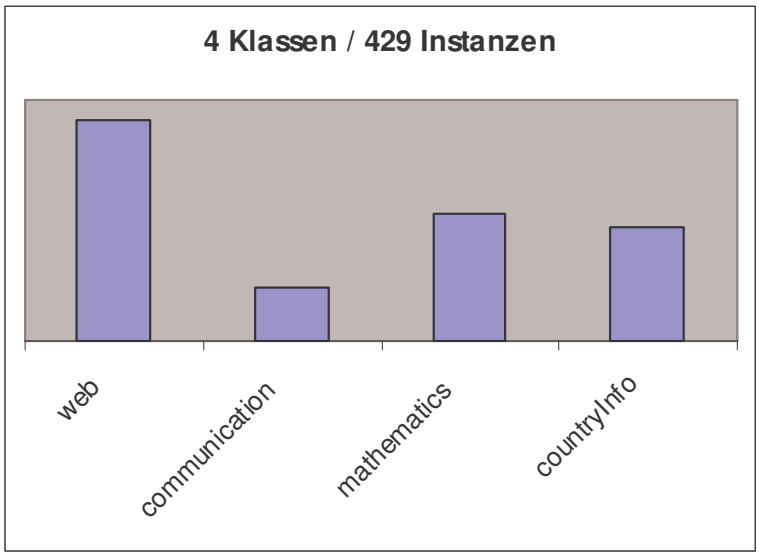
Die markierten Zahlen zeigen die größten Zusammenhänge innerhalb der Korrelationsmatrix auf. Besonders zwischen den Klassen web und developers besteht ein großer Zusammenhang, d.h. wenn man diese beiden Klassen zusammenfasst, werden weniger Instanzen falsch klassifiziert, was insgesamt zu mehr korrekt klassifizierten Instanzen führt.

In Tabelle Abbildung22 sind die fünf Klassen von Untersuchung 5 in vier Klassen vereinfacht und mit der Anzahl der zugehörigen Instanzen, aufgeführt, z.B. gibt es 184 von 429 Instanzen, die Klasse web zugeordnet sind. Dabei stütze ich mich auf die Aufwertung der vorangestellten Korrelationsmatrix.

Abbildung22: 4 Klassen

#	Klassenbezeichnung	enthält Klassen…	Anz
1	web	web & Unterkategorien (36), serverInfo (1), business & Unterkategorien (22), games & Unterkategorien (8), cartoon (2), developers & Unterkategorien (37),databaseprovider (4), datamanagement & Unterkategorien (5),engineering (1), graphics (3), license (1), parser (2), pollcreation (2), usergroups (1), finder & Unterkategorien (46), courierInfo (6), flights & Unterkategorien (5), genbank (1), music (1)	184
2	communication	communication & Unterkategorien (45)	45
3	mathematics	mathematics (10), converter & Unterkategorien (49), money & Unterkategorien (47)	106
4	countryInfo	countryInfo & Unterkategorien (64), news & Unterkategorien (30)	94

Abbildung23: 4 Klassen / 429 Instanzen

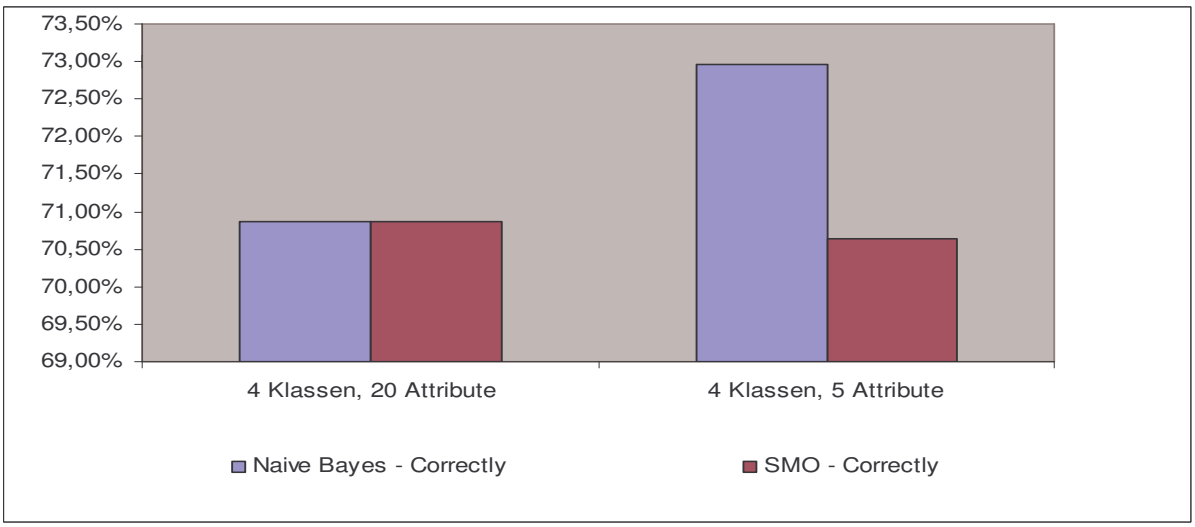


Im Diagramm Abbildung23 sieht man, wie sich die 429 Instanzen der Trainingsmenge auf die vier Klassen verteilen.
Die Klasse web hat die meisten Instanzen. Die Klasse communication hat die wenigsten Instanzen und die Klassen mathematics und countryInfo sind nahezu ausbalanciert.

Das Diagramm Abbildung24 zeigt die Klassifikationsergebnisse für die beiden ARFF-Dateien mit 20 bzw. fünf WSDL-Attributen, eingeteilt in vier Klassen. Verwendet wurden die Lernverfahren Naive Bayes und Support-Vektor-Maschinen, angewendet auf die 429 Instanzen der Trainingsdatenmenge, evaluiert mit zehnfacher Kreuzvalidierung.

Ich stelle fest, dass die ARFF-Datei mit fünf WSDL-Attributen, nur für das Naive-Bayes-Verfahren diesmal bessere Klassifikationsergebnisse liefert, als die ARFF-Datei mit 20 WSDL-Attributen. Das Support-Vektor-Maschinen-Verfahren liefert für beide Datensätze fast homogene Ergebnisse, etwas schlechter mit einer falsch klassifizierten Instanz, schneidet die Klassifikation für die ARFF-Datei mit fünf WSDL-Attributen ab.
Es fällt auf, dass beide Lernverfahren für den ersten Datensatz dieselben Werte für die Klassifikation erreichen, beide Verfahren sind also gleich erfolgreich.
Insgesamt sind die Ergebnisse dieser Klassifikation nochmals besser als bei Untersuchung 5: ein Niveau von ca. 71% korrekt klassifizierter Instanzen bedeutet andererseits, dass nur noch ca. 29% aller Instanzen falsch klassifiziert sind.

Abbildung24: Klassifikation der 4 Klassen



4.8 Untersuchung 7 – Evaluation mit 3 Klassen

Frage: *Lassen sich mit weniger Klassen nochmals bessere Klassifikationsergebnisse erreichen, wie sinnvoll ist das?*

Wie die letzten Untersuchungen zeigen, lassen sich mit einer stetig kleineren Zahl von Klassen, immer bessere Klassifikationsergebnisse erzielen. Jetzt stellt sich die Frage, wie weit man Klassen vereinfachen sollte, um möglichst gute Ergebnisse zu erzielen. Oder bedarf es einer bestimmten Anzahl an Klassen, um eine Menge von WSDL-Dokumenten sinnvollen Klassen zuordnen zu können?

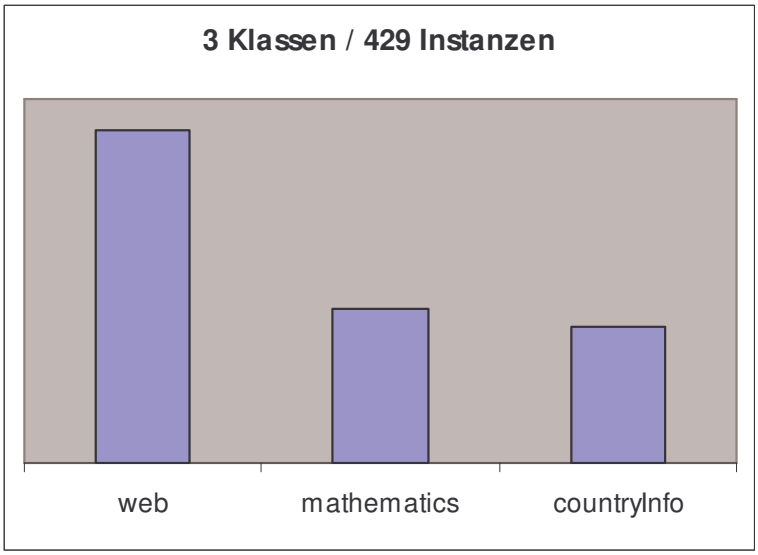
Wenn man sich die Klasseneinteilung aus der 6. Untersuchung veranschaulicht, stellt man fest, dass man inhaltlich nur noch die Klassen `web` und `communication` zusammenfassen kann. Beide Klassen beschreiben einen ähnlichen Bereich: Die Klasse `communication` beschreibt ganz allgemein verschiedene Dienste zur Kommunikation. Bei `web` handelt es sich im weitesten Sinn um die Beschaffung von Informationen über das Internet, man kann also sagen, es handelt sich um eine Kommunikation über das Internet. Die Klassen `countryInfo` und `mathematics` lasse ich unverändert, sie zusammenzufassen macht inhaltlich keinen Sinn. Außerdem waren sie in der letzen Untersuchung zueinander schon ganz gut ausbalanciert.

In Tabelle `Abbildung25` sind die vier Klassen von Untersuchung 6 in drei Klassen vereinfacht und mit der Anzahl der zugehörigen Instanzen aufgeführt, z.B. gibt es 229 von 429 Instanzen, die Klasse `web` zugeordnet sind. Dabei stütze ich mich auf die Überlegung, dass man die Klassen `web` und `communication` inhaltlich zusammenfassen kann.

Abbildung25: `3 Klassen`

#	Klassenbezeichnung	enthält Klassen…	Anz
1	web	web & Unterkategorien (36), serverInfo (1), business & Unterkategorien (22), games & Unterkategorien (8), cartoon (2), developers & Unterkategorien (37),databaseprovider (4), datamanagement & Unterkategorien (5),engineering (1), graphics (3), license (1), parser (2), pollcreation (2), usergroups (1), finder & Unterkategorien (46), courierInfo (6), flights & Unterkategorien (5), genbank (1), music (1), communication & Unterkategorien (45)	229
2	mathematics	mathematics (10), converter & Unterkategorien (49), money & Unterkategorien (47)	106
3	countryInfo	countryInfo & Unterkategorien (64), news & Unterkategorien (30)	94

Abbildung26: 3 Klassen / 429 Instanzen

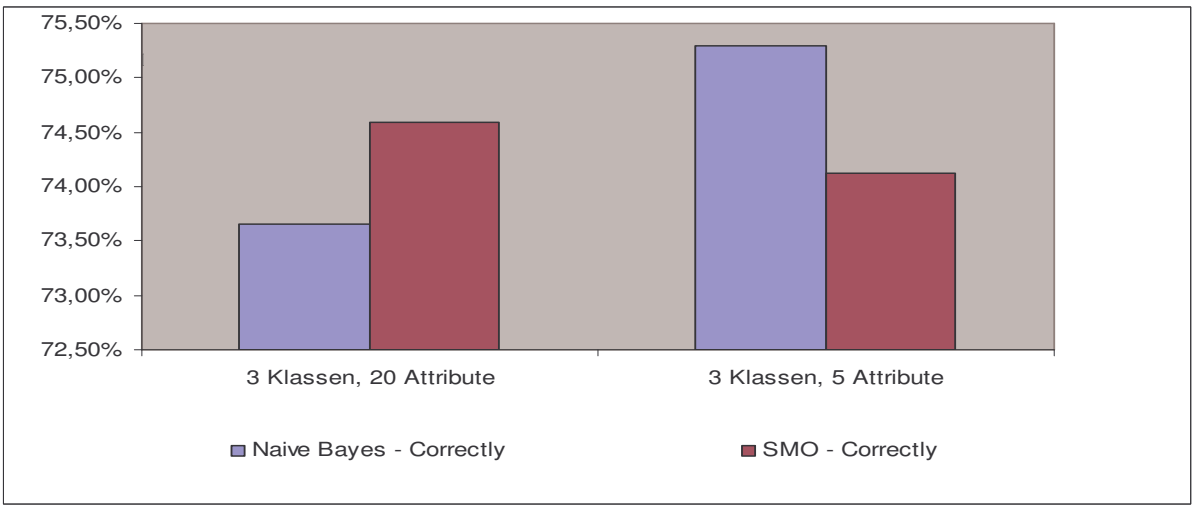


Im Diagramm sieht man, wie sich die 429 Instanzen der Trainingsmenge auf die drei Klassen verteilen.
Die Klassen sind zueinander nochmals besser ausbalanciert. Die Klassen `mathematics` und `countryInfo` beschreiben etwa jeweils ein Viertel der Trainingsdaten, die Klasse `web` beschreibt die restliche Hälfte der Instanzen.

Das Diagramm `Abbildung27` zeigt die Klassifikationsergebnisse für die beiden ARFF-Dateien mit 20 bzw. fünf WSDL-Attributen, eingeteilt in drei Klassen. Verwendet wurden die Lernverfahren Naive Bayes und Support-Vektor-Maschinen, angewendet auf die 429 Instanzen der Trainingsdatenmenge, evaluiert mit zehnfacher Kreuzvalidierung.

Ich stelle fest, dass die ARFF-Datei mit fünf WSDL-Attributen, nur für das Naive-Bayes-Verfahren bessere Klassifikationsergebnisse liefert, als die ARFF-Datei mit 20 WSDL-Attributen. Auffällig in Untersuchung 6 ist, dass das Support-Vektor-Maschine-Verfahren etwas schlechter mit einer falsch klassifizierten Instanz für die modifizierte ARFF-Datei abgeschnitten hat. Diesen Trend findet man auch als Ergebnis dieser Untersuchung, jedoch schneidet hier das Lernverfahren noch schlechter ab, mit zusätzlich zwei falschen Instanzen für die modifizierte ARFF-Datei. Das könnte ein Indiz sein, dass man die Klassen nicht mehr weiter vereinfachen kann, die Grenzen einer sinnvollen Klassifizierung scheinen ausgereizt zu sein. Insgesamt sind die Ergebnisse dieser Klassifikation nochmals besser als bei Untersuchung 6: ein Niveau von ca. 74% korrekt klassifizierter Instanzen bedeutet andererseits, dass nur noch ca. 26% aller Instanzen falsch klassifiziert sind.

Abbildung27: Klassifikation der 3 Klassen



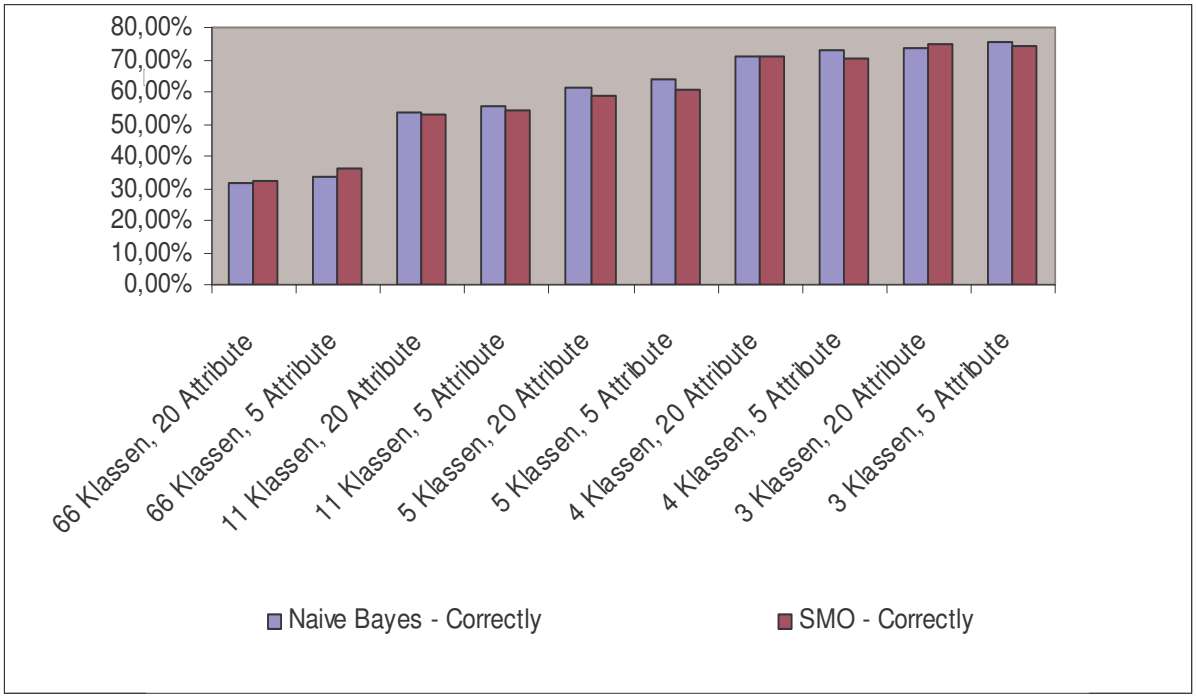
5. Fazit

Als Fazit meiner Ausarbeitung stelle ich fest, das man eine Menge von WSDL-Dokumenten, die man kennzeichnenden Klassen zugeordnet hat, zur Klassifikation von unbekannten WSDL-Dokumenten benutzen kann.

Der Erfolg der Klassifikation von WSDL-Dokumenten hängt entscheidend von der Anzahl der Klassen ab, mit denen man die einzelnen Instanzen der Trainingsmenge klassifiziert. Je mehr Klassen man nutzt, um die begrenzte Anzahl von Instanzen zu beschreiben, desto geringer sind die Aussichten einer erfolgreichen Klassifikation. Andererseits darf man aber auch nicht zu wenige Klassen nutzen, um die begrenzte Anzahl von Instanzen zu beschreiben, da mit abnehmender Zahl von Klassen, die beschreibende Wirkung sich unterscheidender Klassen verloren geht. Man muß einen Mittelweg bei der Einteilung der Instanzen in kennzeichnende Klassen wählen.

Der Erfolg der Klassifikation hängt außerdem von der Auswahl der betrachteten WSDL-Attribute ab, die man zur Klassifikation verwendet. Es zeigte sich, dass man bei der Wahl von fünf bestimmten WSDL-Elemente (siehe Untersuchung 2) nochmals bessere Klassifikationsergebnisse erzielen kann, wenn man die Anzahl der Klassen unverändert lässt.

Abbildung28: Klassifikationsergebnisse



6. Literaturverzeichnis

(Chappell & Jewell, 2002)
Chappel, David A. & Jewell, Tyler (2002):
Java Web Services; O'Reilly Verlag; Köln

(Dostal u.a., 2004)
Dostal, Wolfgang; Jeckle, Mario; Melzer, Ingo; Zengler, Barbara (2004):
Service-orientierte Architekturen mit Web Services: Konzepte-Standards-Praxis;
Spektrum, Akad. Verl.; Heidelberg, München

(Eberhart & Fischer, 2003)
Eberhart, Andreas & Fischer, Stefan (2003):
Web Services – Grundlagen und praktische Umsetzung mit J2EE und .NET;
Carl Hanser Verlag; München, Wien

(Frank, 2005)
Frank, Eibe (2005): Machine Learning with WEKA;
http://puzzle.dl.sourceforge.net/sourceforge/weka/weka.ppt (14.02.2006)

(Hauser & Löwer, 2004)
Hauser, Tobias & Löwer, Ulrich M. (2004):
Web Services – Die Standards;
Galileo Press GmbH; Bonn

(Hess, 2006)
Hess, Andreas (2006):
Collection of Categorized Web Services;
http://www.few.vu.nl/~andreas/projects/annotator/ws2003.html (07.01.2006)

(Kuschke & Wölfel, 2002)
Kuschke, Michael & Wölfel,Ludger (2002):
Web Services kompakt;
Spektrum, Akad. Verl.;Heidelberg, Berlin

(Mertens, 2003)
Mertens, Peter (2003):
XML-Komponenten in der Praxis;
Springer-Verlag;Heidelberg, Berlin

(Szugat, 2005)
Szugat, Martin (2005):
Praktische Einführung in das Data Mining mit WEKA 3.4;
http://bioweka.sourceforge.net/download/LE_1.05_75-79.pdf (14.02.2006)

(Weiss u.a., 2005)
Weiss, Sholom M.; Indurkhya, Nitin; Zhang, Tong; Damerau, Fred J. (2005):
Text Mining – predictive methods for analyzing unstructured information;
Springer-Verlag;New York

(WEKA, 2002)
WEKA-Webseite (2002):
Attribute-Relation File Format (ARFF);
http://www.cs.waikato.ac.nz/~ml/weka/arff.html (07.01.2006)

(Witten & Frank, 2001DE)
Witten, Ian H. & Eibe, Frank (2001):
Data Mining – practical machine learning tools and techniques – german edition;
Carl Hanser Verlag; München, Wien

(Witten & Frank, 2005EN)
Witten, Ian H. & Eibe, Frank (2005):
Data Mining – practical machine learning tools and techniques – second edition;
Morgan Kaufmann Publishers, Inc.; San Fransisco, California