Konzeption und Implementierung einer Schnittstelle für die Eingabe von natürlichsprachlichen Fragen in ein System zur Entdeckung von Wissen in Datenbanken

Praktikumsbericht Technische Informatik

2. Praktisches Studiensemester Sommer 2001

von

Dieter Käppel

angefertigt am

Bayerischen Forschungszentrum für Wissensbasierte Systeme
(FORWISS)
Forschungsgruppe Wissenserwerb
(FG WE)

sowie an der

Georg-Simon-Ohm-Fachhochschule Nürnberg

Betreuer:

Dipl.-Inf. Oliver J. Hogl
(Bayerisches Forschungszentrum für Wissensbasierte Systeme, Erlangen)
Prof. Dr. rer. nat. Peter Gonser
(Georg-Simon-Ohm-Fachhochschule Nürnberg)

Beginn: 01.03.2001
Ende: 18.07.2001
(20 Wochen)

Dieter Käppel
2. Praktisches Studiensemester

Kurzfassung

Das Vorhaben, den Informationsfluss beim Data-Mining zwischen den Domänen-Experten und den interessierten Datenbanken zu optimieren scheitert beim augenblicklichen Stand der Wissenschaft besonders häufig an der Schnittstelle zwischen den natürlichsprachlich formulierten Fragen des Domänen-Experten und der formalisierten Bedienung der Data-Mining-Methoden. Eines der Probleme besteht darin, die variationsreichen natürlichsprachlichen Fragen der Experten in adäquate Aufrufe von Data-Mining-Methoden zu übersetzen. Die vorliegende Arbeit stellt eine erste Benutzerschnittstelle für die Erhebung von solchen Fragen vor. Mit Hilfe von verschiedenen Maßen wird die Ähnlichkeit der eingegeben Fragen zu übersetzbaren Fragen aus der Wissensbasis bewertet. Die höchstbewerteten Kandidaten werden in einem sich anschließenden Feedback-Zyklus dem Benutzer präsentiert, der einen der Vorschläge akzeptiert oder alle verwirft. Im Erfolgsfall wird die gewählte Frage in formalisierter Frage übernommen und kann in einen Data-Mining-Methoden-Ablauf übersetzt werden.

Inhalt

Abbildungsverzeichnis

Formelverzeichnis

Tabellenverzeichnis

1 Einleitung

Im Prozess der Entdeckung von Wissen in Datenbanken gibt es das Problem, dass Endbenutzer, die mit Data-Mining-Methoden ihre Fragen beantwortet haben möchten, im allgemeinen keine Experten in Bezug auf die technische Anwendung dieser Methoden sind. Andererseits ist den Spezialisten für Data-Mining-Methoden meist nicht die Fachsprache der Anwendungsdomäne bekannt. Dem Endbenutzer fällt es schwer, aus der Vielfalt verschiedener Data-Mining-Methoden die für seine Frage passende auszuwählen und zu konfigurieren. Ihm fehlt die Kompetenz, sowohl die Chancen, als auch die Grenzen der verfügbaren Methoden einzuschätzen. Ferner ist der Endanwender häufig überfordert, die von den Data-Mining-Methoden gelieferten Ergebnisse richtig zu interpretieren. Beides erfordert umfassendes Methodenwissen oder die Einbeziehung eines entsprechenden Data-Mining-Experten.

Unter einem Data-Mining-Tool versteht man eine Software, welche ein Interface zu elaborierten Algorithmen aus sehr verschiedenen Bereichen zur Verfügung stellt. Bei vielen Tools des Bereichs Data-Mining, reicht die Programmierung des Interfaces meist nicht sehr viel weiter, als zur bloßen Parameterversorgung der Data-Mining-Methoden. Der Benutzer wird mit den Namen von Fachinformatikern und Fachmathematikern konfrontiert, welche spezielle Algorithmen entwickelt haben. Zumeist haben die Namen der Algorithmen keinerlei Aussagekraft bezüglich deren Funktion. Damit ist der Anwender solcher Tools (z.B. SPSS) dazu gezwungen, sich eingehend mit der Funktion solcher Algorithmen zu beschäftigen, was mit einem sehr hohen Zeitaufwand verbunden ist.

Daneben existieren Experten-Systeme, die sich mehr auf Wissensrepräsentation fixieren. Diese Systeme können natürlichsprachliche Informationen speichern, wiedergeben und können sogar Assoziationen zwischen den Daten herstellen. Experten-Systeme sind jedoch weniger auf die Algorithmik fixiert und bringen einen Domänen-Experten auf dem Weg, komplexe Zusammenhänge zwischen Informationen zu ermitteln, keinen wesentlichen Fortschritt. Die Rechtfertigung für die Forschung besteht darin, fortgeschrittene Data-Mining-Methoden unter Anwendung von Kenntnissen aus dem Bereich der Expertensysteme zu einer natürlichsprachlichen Data-Mining-Anwendung zu verbinden und damit für Domänenexperten und andere informatikferne Personen nutzbar zu machen.

Ziel der Arbeiten in der Forschungsgruppe Wissenserwerb des Bayerischen Forschungszentrums für Wissensbasierte Systeme (FORWISS) ist deshalb ein System, das die Kooperation zwischen beiden Experten adäquat unterstützt bzw. das es erlaubt, Fragen in der Sprache des Endbenutzers zu formulieren, und von den Spezifika des Data Mining weitgehend abstrahieren lässt. Dies kann nur mit einem wissensintensiven Ansatz geleistet werden, da umfangreiches Wissen über einen Anwendungsbereich wie auch umfangreiches Wissen über Data Mining rekonstruiert, repräsentiert und verarbeitet werden muss.

Als Grundlage für die Illustration der Problemstellung dienen im folgenden die seit Dezember 1998 am FORWISS laufenden Studien zum Medizinischen Leistungscontrolling, die in Zusammenarbeit mit der Tiroler Landeskrankenanstalten Ges.m.b.H.[1], Innsbruck durchgeführt wurden. Das Ziel der Studien ist es, vermutete qualitätsrelevante Kriterien für das Medizinische Leistungscontrolling in Patientendaten zu überprüfen und neue Kriterien zu entdecken. In den Studien wurden in den drei Themenbereichen Diagnosen und Therapien, Komplikationen und Dokumentationsqualität Fragestellungen, die von Univ. Doz. Dr. Stühlinger, dem Leiter des Qualitätsmanagements, formuliert wurden, bearbeitet. Grundlage der Auswertungen sind Daten, die in relationaler Form vorliegen. Ein Patient ist unter anderem beschrieben durch Einträge zu Alter, Geschlecht und Geburtsland. Die Attribute einer Behandlung sind im Bereich „Diagnosen“ Hauptdiagnose und Zusatzdiagnosen, im Bereich „Medizinische Leistungen“ Art und Anzahl der Leistungen sowie im Bereich „Verweildauer“

[1] *Die Tiroler Landeskrankenanstalten Ges.m.b.H. (TILAK) ist eine Dienstleistungsgesellschaft zur betriebswirtschaftlich orientierten Unterstützung der Landeskrankenhäuser in Tirol mit Sitz in Innsbruck. Neben anderen Aufgabenbereichen ist vor allem auch Qualitätsmanagement im klinischen Bereich ein wichtiges Dienstleistungsfeld.*

Gesamtverweildauer und Behandelnde Abteilungen. Für eine genauere Beschreibung siehe [Stü00].

Kapitel 2 dieser Arbeit berichtet über die bislang beschrittenen Wege, die Domänen-Experten gehen müssen um Wissen aus Wissensdatenbanken entnehmen zu können. Kapitel 3 beschreibt den „Der Knowledge Discovery Assistant (KDA)“, das wissensbasierte Data-Mining-Assistenzsystem, das derzeit am FORWISS entwickelt wird. Im Kapitel 4 wird beschrieben, mit welchen Mitteln der Autor die Methoden des Wissensmanagements im Bereich Data-Mining verbessern will. Der Fokus zeigt dabei auf eine verständliche Sichtweise, ermangelt jedoch nicht an wissenschaftlichen Ausführungen und Beweisen. Kapitel 5 beschäftigt sich mit den Details der Programmierung und geht konkret auf verwendete Programmiersprachen, Betriebssysteme und Codesequenzen ein. Kapitel 6 betrachtet die Ergebnisse hinsichtlich der Qualität der Implementierung und versucht die durch ein eigens dafür implementiertes Werkzeug messbar zu machen. Schließlich bietet Kapitel 7 eine Zusammenfassung und zukünftige Perspektiven von natürlichsprachlichen Schnittstellen beim Data-Mining. Speziell wird auch auf die Möglichkeiten, welche sich durch eine natürlichsprachliche Kommunikation mit dem Benutzer aufgetan haben, eingegangen.

2 Forschungsstand

Endbenutzer-orientierte Sprachen für die Formulierung von Anfragen an relationale Datenbanken, wie der SQL-Standard (Structured Query Language, [Can92]), und an multidimensionale Datenbanken, wie OLAP (On-Line Analytical Processing,[Ber97]), haben sich im Bereich der Datenbanksysteme etabliert. Die Ergebnisse dieser Art von Anfragen sind jedoch stets nur Datensätze oder Fallmengen. Für die Lieferung von Mengen von Aussagen oder Beschreibungen komplexerer Muster wurden DMQL (Data Mining Query Language,[Han96]) und verschiedene Data-Mining-Template-Sprachen (z.B.[Kle94]) entwickelt. Diese Sprachen erfordern vom Benutzer jedoch Hintergrundwissen über Data-Mining- und Datenbank-Konzepte und sind somit nur begrenzt für Endbenutzer geeignet.

Aus diesem Grund gibt es bereits Ansätze, Endbenutzern bei der Formulierung von Anfragen an Informationssysteme Hilfe anzubieten, die kein Wissen über technische Strukturen erfordern. Im Bereich der Datenbanken wurden über SQL hinausgehende Vorschläge gemacht, um Anfragen in der Fachsprache von Endbenutzern zu repräsentieren, die Natural Language Interfaces to Databases (NLIDB).

2.1 Geschichte von natürlichsprachlichen Anfragessytsemen

I. Androutsopoulos, G. D. Ritchie und P. Tanisch geben in [And94] eine Kurzeinführung in diese natürlichsprachliche Schnittstellen für Datenbanken: Einer der ersten Versuche, natürlichsprachliche Wissensdatenbanken zu implementieren, wird in [Woo72] beschrieben. Es handelt sich dabei um die Realisierung einer speziellen Datenbank mit nur geringer Abstraktion bezüglich der Fragen. Daraufhin erschienen in den Siebzigern einige weitere Datenbanken mit natürlichsprachlichen Schnittstellen. RENDEZVOUS von [Cod74] implementiert Benutzerdialoge mit einem Hilfesystem zur Formulierung von Fragen. LADDER (in [Hen78]) konnte durch die Verwendung semantischer Grammatiken bereits auf große Datenbanken angewandt sowie für verschiedene Datenbanken konfiguriert werden. Es handelt sich dabei um eine Technik, welche Syntax- und Semantikverarbeitung miteinander verbindet. Trotz semantischer Grammatiken konnten die resultierenden Systeme nur schwer auf andere Themengebiete übertragen werden. Als Forscher begannen, portierbare Systeme zu fokussieren, wurde die Verwendung semantischer Grammatiken wieder stufenweise reduziert. Es entstanden Systeme wie PLANES und PHILQA1, welche in [Wal78] und [Sch77] beschrieben werden. Wie in [War82] illustriert, entstand Anfang der achtziger Jahre unter anderem eines der damals am weitesten verbreitetsten Systeme, CHAT-80, dessen Code weit verbreitet wurde und vielfach zu neuen natürlichsprachlichen Schnittstellen weiterverarbeitet wurde wie z.B. [Aux86] oder [And93] beschreiben.

Die Forschungen in der Mitte der achtziger Jahre brachten reichhaltige Ergebnisse und man implementierte zahlreiche Prototypen, wie in [Gro83], [Gro87] und [Mar86] nachzulesen ist. Man machte Portierbarkeit zu einem zentralen Angelpunkt, während die einfache Konfigurierbarkeit für Datenbankadministratoren geplant wurde. In [Joh85] wird vorgeschlagen, dass bis 1987 natürlichsprachliche Schnittstellen als Standard für Wissenssysteme gelten sollten.

In den folgenden Jahren kann man einen signifikanten Rückgang der Veröffentlichungen zu diesem Thema feststellen und natürlichsprachliche Interfaces entwickelten sich zu einem Außenseiterthema. Das konnte einige Wissenschaftler jedoch nicht davon abhalten, die Schnittstellen dennoch weiter zu entwickeln und Fortschritte aus allgemeinen natürlichsprachlichen Disziplinen mit einzuarbeiten. Aus diesen generellen natürlichsprachlichen Parsern, welche mit zusätzlichen Modulen zur Konstruktion komplexer Abfragen mit Datenbanken verbunden wurden, entwickelten sich kommerzielle Produkte wie INTELLECT [Har84], BBN´S PARLANCE [Bat86], IBM´S LANGUAGEACCESS [Ott92], Q&A von Symantec, NATURAL LANGUAGE [Cop90], LOQUI [Bin91] und ENGLISH WIZARD von Linguistic Technology Corporation. Einige zusammenfassende Eigenschaften darüber können in [Sij93] nachgelesen werden.

Natürlich entwickelten sich in dieser Zeit nicht nur die Front-Ends weiter, sondern auch die Datenbanken selbst. Codd in [Cod70] hatte maßgeblichen Einfluss auf die Entwicklung der Systeme wie sie uns heute bekannt sind. Obwohl die Entwicklung von SQL ein entscheiden-

der Schritt in der Entwicklung der formalen Sprachen darstellte, konnten dadurch die natürlichsprachlichen Schnittstellen trotzdem nicht verdrängt werden. Die Domänen-Experten waren weiterhin wenig daran interessiert, formale – wenn auch vereinfachte – Sprachen zu erlernen. Die mangelnde Akzeptanz der natürlichsprachlichen Schnittstellen bestand darin, dass die Künstliche Intelligenz weiterhin nur eine Teilmenge der tatsächlichen linguistischen Komplexität transformieren konnte. Der Benutzer war somit gezwungen, herauszufinden, welche Teilmenge das System verstand, um seine Fragen dementsprechend umzuformulieren, so dass diese vom System korrekt interpretiert wurden. Das erdrückende Argument war nicht, dass es leichter war, eine formale Sprache zu erlernen, als vielmehr dass man sagte, das Erkennen der interpretierbaren Teilmenge der natürlichen Sprache sei eine Art des Formalisierens und damit handle es sich um keine natürliche Sprache mehr.

Weiter wird kritisiert, dass die linguistischen Fähigkeiten des Systems für den Anwender kaum greifbar seien. In der Analyse natürlichsprachlicher Schnittstellen fing man an, unterschiedliche Arten des Fehlverhaltens zu differenzieren. Erstens orientierte man sich an Irrtümern in der Erwartungsbildung seitens des Benutzers. Man unterscheidet zwischen „falschen positiven Erwartungen“ und „falschen negativen Erwartungen“. Bei einer falschen positiven Erwartung geht der Benutzer davon aus, das System verstünde seine Frage, was zu einer falschen Antwort seitens des Systems führen kann, wohingegen bei der falschen negativen Erwartung der Benutzer davon ausgeht, das System verstehe seine Frage nicht, worauf hin er sie unnötig umformuliert.

Zweitens führte man als Gütemaß eines natürlichsprachlichen Systems die Unterscheidung zwischen linguistischen und konzeptionellen Fehler ein. Ein linguistischer Fehler entsteht durch Fehl- oder Nichtinterpretation einer Frage seitens der natürlichsprachlichen Schnittstelle, dagegen ist die Ursache eines konzeptionellen Fehlers in der Unvollkommenheit der Schnittstelle zwischen Linguistik und Datenbank zu suchen, d.h. man sah diesen Fragetyp während der Konzeption des Systems nicht vor. Versuche wurden unternommen, dem Benutzer Rückmeldungen darüber zu vermitteln und man stellte fest, dass diese Angelegenheit für das System mindestens genauso diffizil ist, wie für den Benutzer festzustellen, ob das System seine Frage korrekt interpretiert hat.

Eine dritte Quelle für Fehler liegt in der Intelligenz der natürlichsprachlichen Schnittstelle, welche dem Anwender suggeriert wird. Für gewöhnlich ist ein derartiges System weder zur Deduktion fähig, noch besitzt es Eigenschaften, welche wir einem Gesprächspartner für gewöhnlich als gemeine geistige Fähigkeiten unterstellen. Grafische oder formale Systeme verführen den Benutzer weniger zu solchen Annahmen.

Als vierte Ursache unterstellt man, die natürliche Sprache sei ein unpassendes Medium für eine Kommunikation zwischen Mensch und Computer wegen ihrer Redundanzen und Ambiguitäten bezüglich Kontext, Junktionen oder Ellipsen. Grafische und formalbasierte Schnittstellen sind aufgrund ihrer axiomatischen Grundlage eindeutig und größtenteils frei von Redundanzen. Teilweise bekommt man diese Probleme durch Heuristiken „diskursive Erwartungen“ und „Kontextersetzung“ in den Griff. Über Beseitigung von Ambiguitäten sei ein Exkurs in [Per88] empfohlen.

Als fünftes grundliegendes Problem ernannte man das Konfigurationsproblem. Während formale Schnittstellen weitgehend automatisiert sind, d.h. konfigurationsfrei verwendet werden können, müssen natürlichsprachliche Schnittstellen erst eine aufwendige Konfiguration erfahren, bevor sie praktisch einsatzfähig sind. Als Folge davon verlagert sich der Schwerpunkt des Nutzens natürlichsprachlicher Schnittstellen weg von einfachen Abfragen hin zu komplexen Abfragen, welche durch Kombination der Konfigurationsmenge definiert sind, wie in [Jar85] und [Sma83] nachzulesen ist.

2.2 Architekturen

Aus der Historie geht bereits hervor, dass verschiedene Architekturen für natürlichsprachliche Interfaces verwendet worden sind, wobei alle bestimmte Vor- sowie Nachteile beherbergen.

2.2.1 Keyword Spotting

Eine naheliegende Möglichkeit der Implementation sind einfache Mustererkennungssysteme, die Ähnlichkeiten aus den Zusammensetzungen der Benutzerfrage und gespeicherten Fragen für die Sprachanalyse gebraucht. Außer den Mustererkennungsalgorithmen stellt diese Form von Architektur den kleinsten Aufwand von allen Architekturen dar, denn es braucht keine Form von Parsing implementiert zu werden wie in [Joh85] nachzulesen ist.

Ein einfaches Mustererkennungssystem lässt sich beispielsweise durch Regeln erzeugen, wie im Folgenden gezeigt wird:

Muster: ... „Diagnose“ ... <Aufenthaltsdauer>
Übersetzung: Zeige Diagnose aus den Datensätzen mit Aufenthaltsdauer = <Aufenthaltsdauer>

Muster: ... „Diagnose“ ... „Aufenthaltsdauer“
Übersetzung: Zeige Diagnose und Aufenthaltsdauer aus den Datensätzen.

Es soll nicht verschwiegen werden, dass solche Architekturen sehr fehleranfällig für Ambiguitäten sind, denn häufig haben Wörter mit nur leicht unterschiedlicher Schreibweise einen vollkommen anderen Wortstamm zugrunde liegen. Die mangelnde Interpretation der Frageninhalte zeigt sich als weiterer Makel, welche einen praktischen Einsatz vereiteln.

2.2.2 Syntaxanalyse

Eine andere Idee ist, die natürlichsprachliche Eingabe als syntaktisches Muster zu sehen und mittels Syntaxparsern zu analysieren, wie auch [Woo72] schreibt. Mit einer vollständigen Syntaxdefinition z.B. der deutschen Sprache wären die Fehlinterpretationen von Mustererkennungsalgorithmen zwar ausgemerzt, doch bringt diese Idee wieder die Nachteile mit sich, dass die menschliche Sprache ein sehr komplexes Medium darstellt mit vielen Wörtern, Flexionen und Ausnahmen, welches einem ständigen Wandel und lokalen Anomalien unterliegt, wobei sich die Lokalität aus Region und Domäne zusammensetzt. Dabei sind bei langfristig geplanten Systemen auch die zeitlichen Änderungen der Sprache im Auge zu behalten. Beschränkte man sich auf eine lokale Sprache, so müsste man dennoch eine sehr komplexe Grammatik implementieren, um alleine bei den Wortbeugungsformen bezüglich Numerus, Genus, Kasus, Modus und Tempus ein Erkennen zu ermöglichen.

Nicht zuletzt im Zuge der Internationalisierung ergeben sich noch weitere Probleme. So kann ein syntaxbasiertes System durch den bloßen Austausch der Wortstämme und Flexionsregeln nicht notwendigerweise an neue Sprachen angepasst werden.

2.2.3 Semantische Grammatiksysteme

Semantische Grammatiksysteme bauen auf die Theorie der Morpheme auf, deren Wissenschaft als Morphologie bezeichnet wird. Die Etymologie des Wortes Morphem geht auf das Griechische zurück und bedeutet soviel wie Form, Gestalt. In der Morphologie fasst man ein Wort als komplexe Entität auf, welche im Zusammenhang mit der Phonologie, Grammatik (Morphologie und Syntax) und der Semantik eine bestimmte Einheit anzeigt. Da sich die Diskussion um die Morphologie noch in einer Kontroverse befindet, sind verschiedene Definitionen wie „Wesen und Gegenstand der Grammatik“, „Lehre von den Wortarten“ und „Die Lehre von den morphologischen Kategorien“ bezieht sich auf übergeordnete logische Begriffe, welche die Grundrelationen des Seins beschreiben, wie z.B. Raum, Zeit, Qualität, Quantität, Kausalität, Bewegung und Materie.

Aufbauend auf diese Definitionen und Denkansätze sind semantische Grammatiksysteme entstanden, welche eine Möglichkeit darstellen, festes Domänenwissen einfach in andere Systeme einzubinden. Zur Konzeptualisierung der morphologischen Analyse verwendet man Kategorialgrammatiken, Phrasenstrukturgrammatiken oder Linksassoziative Grammatiken, wobei die letzten den interessanten Aspekt beherbergen, die Morphologie von links nach rechts analog dem Menschen aufzufassen. Als einzelne Segmente betrachtet man bei dieser Form der oberflächenkompositionalen Analyse die Allomorphe. Unter einem Allomorph versteht man die konkrete Realisation eines Morphems wie z.B. be-, ge- und –s. Die Allomorphe

können u.a. durch komplementäre Ausschließungen eine grammatikalische Korrektheit definieren. Während der Analyse wird ständig ein Allomorphlexikon zeichenweise auf den noch nicht abgeleiteten Wortformteil angewandt, bis der Algorithmus entweder an das Ende des Wortes stößt oder erfolglos terminiert.

Ein Vorteil gegenüber Keyword Spotting und den Syntaxparsern besteht in der Gewichtung bestimmter Merkmale durch Wahrscheinlichkeiten, welche man dann während eines zweiten Durchlaufs zur Feststellung der Wortart und Flexionsform verwenden kann. Zusätzlich lässt sich die halbautomatische Eingliederung unbekannter Wortformen in ein bestehendes Lexikon erreichen. Eine Disambiguierung kann anhand von Entscheidungskriterien wie die einfache Häufigkeit, die Verbundhäufigkeit und die Gesamtzahl der vorkommenden Allomorphe vorgenommen werden, wie [Hau98] beschreibt. Eine weitere deutliche Verbesserung ist auch die Decodierung der Affixe wie Präfixe, Zirkumfixe und Suffixe von welchen gerade die Fachsprache häufig Gebrauch macht. Die Repräsentation der fertigen Semantik erfolgt dabei häufig durch Semantik- oder Parsbäume, deren Ordnung durch sinkende Intensität der semantischen Bedeutung von der Wurzel bis zu den Blättern festgelegt ist. Damit ist es weiterverarbeitenden Programmen möglich auf die im entsprechenden Satz enthaltenen Informationen strukturiert zuzugreifen.

2.2.4 Mehrschichtsysteme

Mittels Mehrschichtsystemen will man noch einen Schritt weitergehen und die Sprache nicht nur kontextfrei analysieren, sondern die Möglichkeit schaffen, durch Einbindung einer dynamischen Zwischenschicht kommunikative Mikro- und Makroziele, Partnermodelle (Überzeugungen, Wissensstände), Weltwissen, emotionale Zustände, Linearisierung (topologische Ordnung einer Fahrtroute) zu implementieren. Beobachtung bzw. Vergleich der synthetischen Antworten mit den Antworten des Partners stellen zusätzliche Möglichkeiten dar.

3 Der Knowledge Discovery Assistant (KDA)

Eine Software, welche Data-Mining für Domänen-Experten zugänglich macht, die sich nicht eindringlich mit der Benutzung von Statistik-Software und schwierigen mathematischen Methoden befasst haben, wird seit einiger Zeit am Bayrischen Forschungszentrum für Wissensbasierte Systeme (FORWISS) entwickelt und Knowledge Discovery Assistant (KDA) bezeichnet.

Diese Software wird bislang vordringlich im Bereich des medizinischen Qualitätsmanagements eingesetzt, um spezielle Fragen z.B. über das Auftreten von Komplikationen und die Bewertung der Dokumentationsqualität in Krankenhäusern zu klären. Sie befindet sich seit dem Jahr 1997 in der Entwicklung und besteht aus mehreren Komponenten, welche zum Teil im Rahmen von Diplomarbeiten bzw. Dissertationen angefertigt wurden. Die Software ist modular aufgebaut, um schnell neuen Bedürfnissen und einem erweiterten Erkenntnisstand angepasst zu werden:

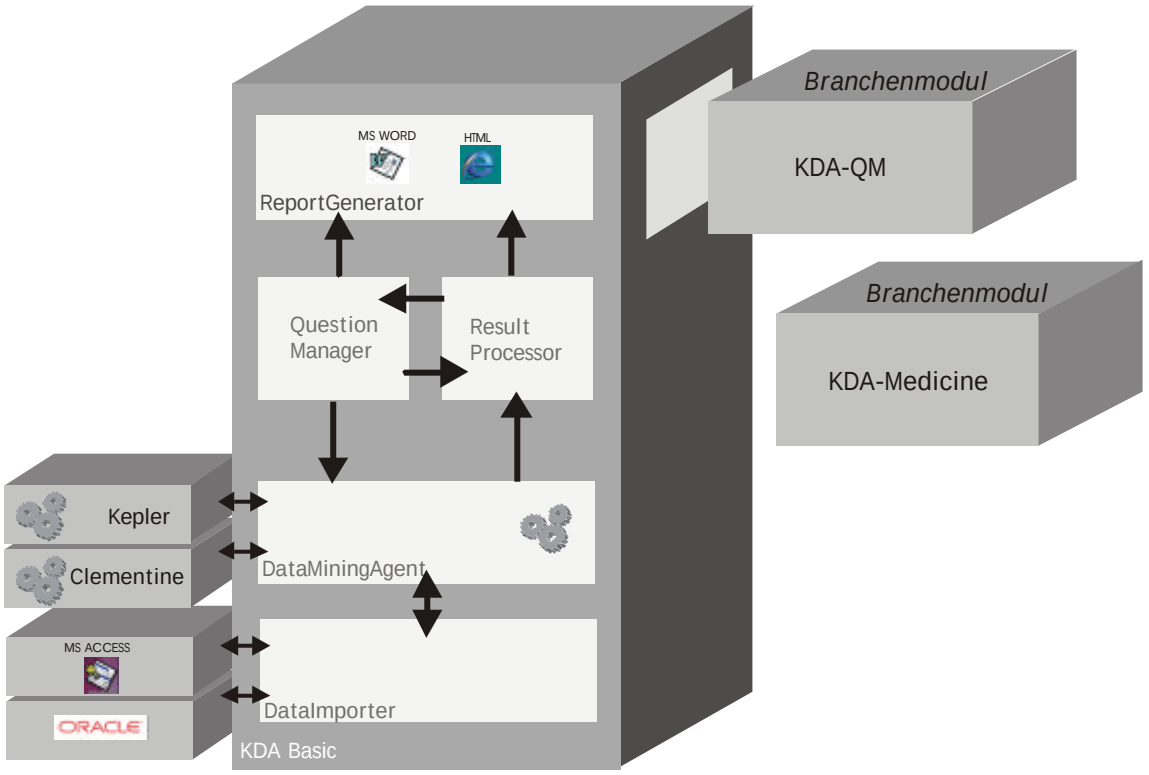

Abbildung 1 - KDA Konzept

Der KDA besteht aus mehreren Komponenten, wie in der Abbildung 1 - KDA Konzept dargestellt wird:

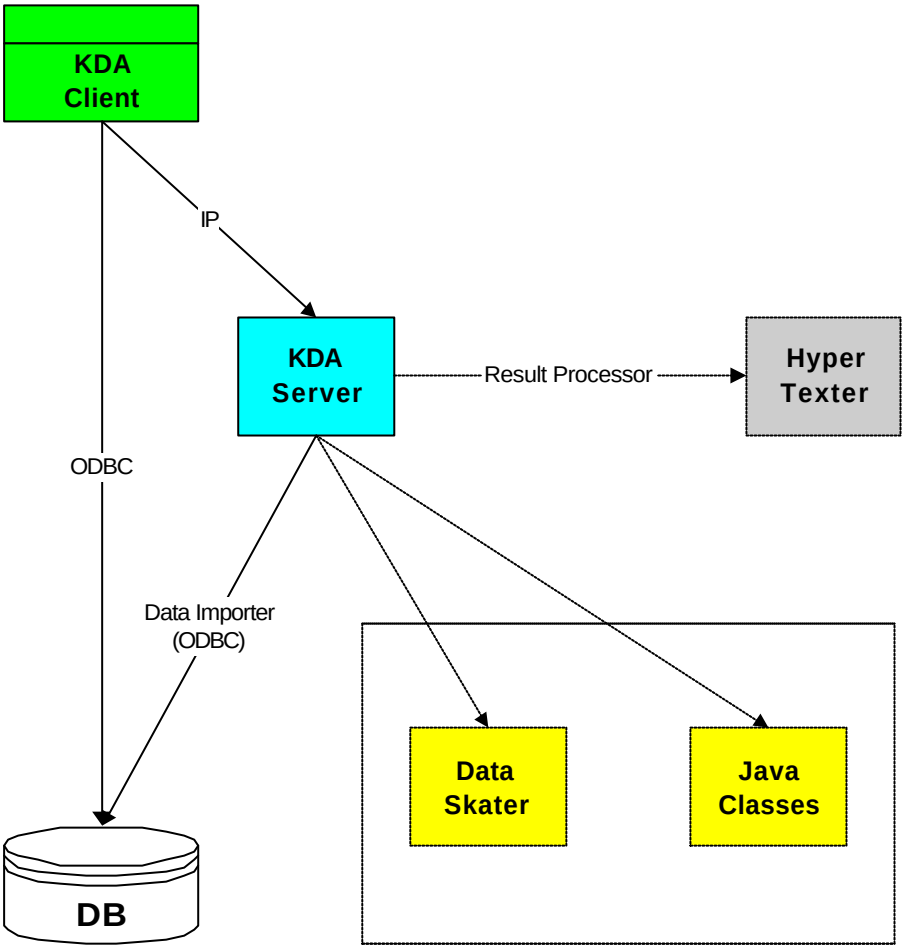

Abbildung 2 - KDA Komponenten

Es ist eine ganze Reihe von Softwarekomponenten nötig, um diese Klasse von Problemen zu lösen. Als Grundlage zur Problemlösung dient eine Aufteilung der Software in drei Abstraktionsebenen.

3.1 Abstraktionsebenen des KDA-Modells

Ref. Auf Zeichnung

3.1.1 KD-Schicht

An oberster Stelle sitzt die sogenannte KD-Schicht, welche die Interaktion mit dem Benutzer abwickelt. Diese Schicht nimmt die Fragen des Benutzers entgegen und präsentiert ihm die entsprechenden Antworten. Für die Formulierung der Fragen des Benutzers wird die am FORWISS entwickelte *Knowledge Discovery Question Language* (KDQL, [Hog01]) verwendet. KDQL ist ein Ansatz für eine kontrollierte Sprache, die leicht von Fachexperten benutzt und verstanden werden kann. Dafür wurden die Strukturen von natürlichsprachlichen Fragen übernommen und in eine semantische Grammatik abgebildet. Im Folgenden wird die Grundstruktur von KDQL in einer erweiterten BNF-Grammatik definiert. Optionale Elemente sind darin mit „[]“ gekennzeichnet.

⟨KD-Frage⟩ ::=
⟨FrageTyp⟩
[⟨FrageGruppe⟩]
[⟨FrageKontext⟩] ?

In KDQL bestimmt der "⟨FrageTyp⟩" den Typ möglicher Antworten. Unterschiedliche Fragetypen können unterschiedliche Frageobjekte und Frageargumente zur Folge haben, so dass diese abhängig vom Fragetyp modelliert werden müssen. Das optionale Konzept „⟨Frage-Gruppe⟩" erlaubt dem Benutzer die Bildung von vergleichbaren Gruppierungen der Daten. Das Konzept "⟨FrageKontext⟩" ermöglicht die Auswahl einer eingeschränkten Sicht auf die Gesamtdaten.

Es sind zusätzlich zwei Schichten notwendig, um die Fragen des Domänen-Experten beantworten zu können.

3.1.2 DM-Schicht

In der DM-Schicht befinden sich die eigentlichen Data-Mining-Methoden. Es handelt sich dabei um Algorithmen wie Parameterschätzverfahren, Neuronale Netze, Korrelation, Regression, Verfahren des Maschinellen Lernens, Entscheidungsbaum- und Regelinduktion, Modellprognose, Prädiktion und andere Algorithmen.

3.1.3 DB-Schicht

In der untersten Schicht, der DB-Schicht, befindet sich schließlich die Datenbank, in der die Informationen gespeichert sind, auf denen die Data-Mining-Algorithmen arbeiten. Es handelt sich dabei um in verschiedenen Krankenhäusern dokumentierte Patientenfälle etc., die in einer relationalen Datenbank gespeichert sind.

3.1.4 Zusammenfassung

Veranschaulicht man die Abstraktionsebenen in der „Abbildung 3 – Abstraktionsebenen" erkennt man den Informationsfluss während des Data-Mining-Prozesses.

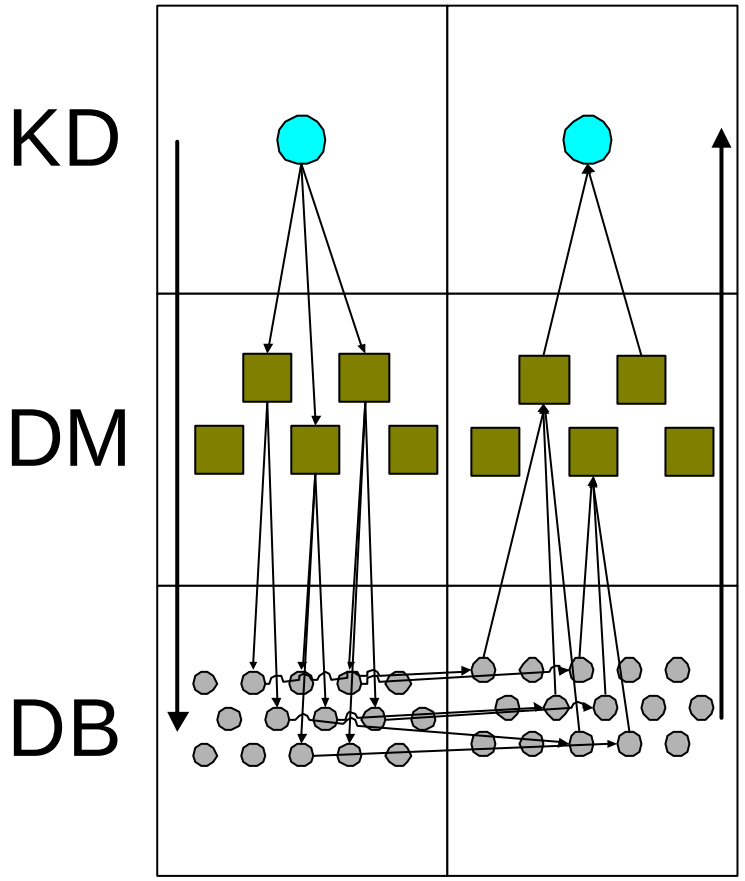

Abbildung 3 – Abstraktionsebenen

In Abbildung 3 – Abstraktionsebenen ist der Zyklus des Data-Mining zu erkennen. In der KD-Schicht befinden sich als Kreise dargestellt die KD-Fragen, welche innerhalb dieser Schicht zur Konkretisierung noch weiter aufgespaltet werden können. So kann eine KD-Frage zu mehreren Data-Mining-Aufrufen expandieren. Jeder Data-Mining-Aufruf, durch Quadrate in der Abbildung dargestellt, kann nun weitere Datensätze aus der Datenbank abrufen. Die abgefragten Datensätze werden in der DM-Schicht je nach ausgewählten Algorithmus auf sehr verschiedene Weise verdichtet und zu den Antworten der konkretisierten Fragen der KD-Schicht modelliert. Aus der Hierarchie des Fragenmodells kann die KD-Schicht die ursprüngliche, abstrakte Frage beantworten.

Trotz des sehr stark vereinfachten Modells ist sehr deutlich zu erkennen, dass durch eine einfache Frage unter Umständen mehrere Data-Mining-Methoden zur Rate gezogen werden müssen, welche dann auf eine größere Menge von Daten aus der relationalen Datenbank zugreifen. Durch geeignete Abbildungen werden die Daten von den Datenbank-Anfragen wieder verdichtet, um die Data-Mining-Methoden mit Informationen versorgen zu können. Die von den Data-Mining-Methoden gelieferten Ergebnisse werden weiter verdichtet, um schließlich die Frage aus der KD-Ebene zu beantworten.

Das Modell ist deswegen stark vereinfacht, weil die Frage nicht wie sie vom Experten eingegeben wurde sofort von der KD-Ebene an die DM-Ebene übertragen werden kann. Im Fall, dass die KD-Ebene die Frage bereits richtig interpretiert hat, muss diese noch immer aufgespaltet werden durch den sog. Prozess der Expansion bzw. Konkretisierung. Abstrakte Fragеelemente wie z.B. gruppierte Objekte müssen in ihre semantischen Komponenten zerlegt werden, damit die DM-Algorithmen mit Informationen über die Datenbank versorgt werden können.

Die Datenkommunikation zur relationalen Datenbank soll hier nur insofern betrachtet werden, als dass dies interessant ist für die Schnittstelle zur Natürlichsprachlichen Eingabe. Der Fokus dieser Arbeit richtet sich auf die Erzeugung abstrakter Fragen, was zur Abbildung 4 - Informationsfluss bei der Fragenerzeugung, einer genaueren Betrachtung des Informationsflusses bis zu den Data-Mining-Methoden führt:

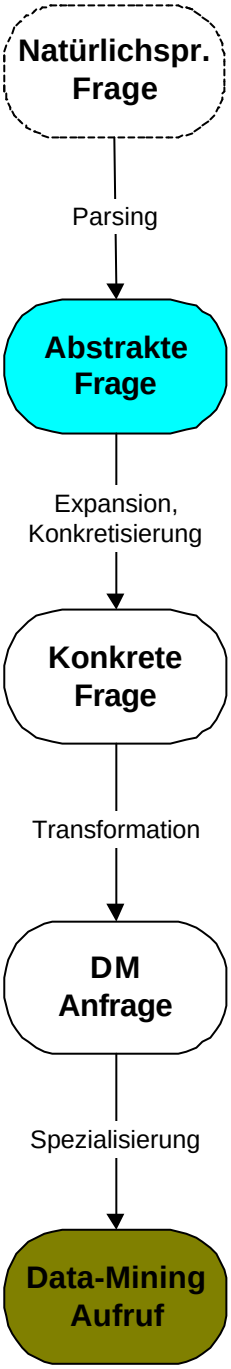

Abbildung 4 - Informationsfluss bei der Fragenerzeugung

In Abbildung 4 - Informationsfluss bei der Fragenerzeugung ist die Stelle, an welcher das natürlichsprachliche Interface angebunden werden muss graphisch dargestellt. Dabei wurde auf die bildliche Darstellung der Rücktransformation der Antwort verzichtet.

4 Konzeption der Frageneingabe und -analyse

Aufgrund der Ermangelung der bisherigen Data-Mining Anwendungen, natürlichsprachliche Anfragen an das Wissenssystem zu verarbeiten, besteht in erster Linie die Notwendigkeit eine Möglichkeit zu schaffen, damit Domänen-Experten die Fragen freier an das System richten können. Mit „freier“ wird hier der Bezug zu vor der Fragestellung definierten Prämissen – und damit auch Frage-Mustern – verstanden.

Das bisherige KDA-System hat zwar schon den Schritt gemacht, natürlichsprachliche Fragen konstruktiv zu bearbeiten, die Einschränkung besteht jedoch immer noch darin, dass der Domänen-Experte die konstruktiven Funktionen nur mit Fachkenntnissen über deren Konsistenz benutzen kann. Gerade das ist zu vermeiden. Der erste Schritt dafür, ähnliche Sätze erkennen zu können. Aus einem Vorrat an nach absteigender Relevanz sortierten Satzmustern kann der Experte den ihn am besten passenden auswählen. Man erzeugt diese Satzmuster dynamisch aus Satzfragmenten, um ein zu starres Verhalten des Systems zu vermeiden.

4.1 Dynamischer Mustervorrat

Weil natürliche Sprachen eine Entwicklungsgeschichte haben, können die gültigen Sätze nicht durch Logik alleine beschrieben werden. Z.B. folgen deutsche Sätze anderen Mustern wie z.B. die Sätze der englischen Sprache. Infolgedessen ist es erforderlich, einen Mustervorrat an gültigen Sätzen anzulegen, bei nach der Überprüfung der Relevanz verschiedene Komponenten aus einer Datenbank eingesetzt werden können. Diese Muster sollen eine möglichst vollständige Erkennung der Fragesätze, welche für den Anwender der Anwendung relevant sind, umschließen.

4.2 Fragenrelevanz

Für die Evaluierung einer Relevanz ist eine linguistische Metrik notwendig, welche die Ähnlichkeit zweier Sätze bestimmt. Ein Algorithmus soll für jeden im Mustervorrat enthaltenen Fragesatz einen Abstand zu dem vom Benutzer eingegebenen Fragesatz errechnen, um durch einen Relevanzwert eine absteigend sortierte Liste zu erstellen.

Ein dafür in Frage kommender Algorithmus kann mit einer aus der quadratischen Optimierung bekannten Methode, der sogenannten Weighted-Least-Mean-Square-Methode, wie er auch in der Regressionsanalyse Verwendung findet, arbeiten. Als Optimierungskriterien kommen die Abstände ähnlicher Morpheme untereinander sowie die Abstände ähnlicher Wörter untereinander in Frage. Die Verschachtelung der beiden Optimierungskriterien kann dabei z.B. über das maximale Zeilensummenkriterium erfolgen um eine Konvergenz zu garantieren. Auf die Forderung nach der Minimalität des Abstands wird verzichtet, d.h. es wird nicht nur der vom Algorithmus für am besten passende empfundene Satz, sondern stattdessen auch weniger optimale Fragesätze mit steigendem Abstand, also sinkender Relevanz angezeigt.

Die Idee dabei ist, zuerst eine Metrik für Wörter anzuwenden. Ausgehend von der einfachen Heuristik, Wörter sind sich je ähnlicher, je mehr ähnliche Morpheme sich an nahe beieinander liegenden Stellen befinden, definiert man die Funktion

$$Q(u,v)=\begin{cases}1 \text{ für } u=v\\ 0 \text{ für } u\neq v\end{cases}$$

Gleichung 1 - Einfache Ähnlichkeitsfunktion für Morpheme

Beispiel 1. Verwendet man in vereinfachter Weise als Morpheme genau die Buchstaben des Wortes, so ist $Q('s','s')=1$ und $Q('s','a')=0$. ♦

Man berechnet den Abstand zweier Wörter (Morphemvektoren) a und b durch

$$d_{wort}(a,b) = \sum_{j=1}^{m} \min_{i=1}^{n} \left((i-j)^2 + a\left(1 - Q(a_i, b_j)\right)\right)$$ **Gleichung 2 - Rekursive Kombination**

wobei a einen zusätzlichen Parameter darstellt, um das Verhalten des Algorithmus zu beeinflussen. Wählt man a größer, so verringert sich der Einfluss des Abstands gleicher Morpheme im Abstand zweier Wörter. In der Praxis hat sich gezeigt, dass $a = 20$ eine gute Wahl für den Parameter darstellt.

Beispiel 2. Sei $a = 'nehme'$, $b = 'nahm'$ und $a = 20$. Die Funktion $d_{wort}(a,b)$ kann man durch eine Tabelle darstellen:

	‚n', i=1	‚e', i=2	‚h', i=3	‚m', i=4	‚e', i=5	Min
‚n', j=1	0	21	24	29	36	0
‚a', j=2	21	20	12	24	29	12
‚h', j=3	24	21	0	21	24	0
‚m', j=4	29	24	21	0	21	0
Summe						12

Tabelle 1 - Wortabstand

Die Vorgehensweise zur Berechnung des entgültigen Wertes erfolgt in drei Schritten. Im ersten Schritt berechnet man an allen Kreuzungsstellen für die Morpheme (d.h. alle Kombinationen der Morpheme beider Wörter) der beiden Wörter (hier die Buchstaben) die Werte durch das Quadrat der Differenz zwischen i und j plus 3, wenn die Morpheme an dieser Stelle ungleich sind. Im zweiten Schritt berechnet man das Minimum jeder Zeile und bildet im dritten Schritt die Summe der Minima und erhält damit schließlich den gewünschten Wert. ♦

Man kann jetzt auch den Abstand zwischen den Vektoren A und B, welche Vektoren aus Morphemvektoren darstellen, berechnen. Man bildet wieder analog zu den Wörtern die Summe der Minima, jedoch addiert man noch den Wert eins zum Wortabstand, um eine Auslöschung der Wortähnlichkeit zu vermeiden. Mit

$$d_{satz}(A,B) = \sum_{j=1}^{m} \min_{i=1}^{n} \left((i-j)^2 + b \right) d_{wort}(A_i, B_j)$$ **Gleichung 3 - Kombination von Wörtern**

Erhält man die gewünschten Werte. Die Praxis hat gezeigt, dass $b = 0.8$ eine gute Wahl für den Parameter darstellt.

Beispiel 3. Sei $A = ('ein', 'test')$, $B = ('einmal', 'testen')$, $C = ('ganz', 'anders')$ und $b = 1$. Die Funktion $d_{wort}(A_i, B_j)$ bzw. $d_{wort}(A_i, C_j)$ berechnet man analog Beispiel 2 und erhält die Tabellen:

$\left((i-j)^2 + 1\right) \cdot d_{wort}(A_i, B_j)$	‚ein', i=1	‚test', i=2	Min
‚einmal', j=1	1•74	2•106	74
‚testen', j=2	2•87	1•33	33
Summe			107

Tabelle 2 - Satzabstand A-B

$\left((i-j)^2+1\right)\cdot d_{wort}\left(A_i,C_j\right)$	‚ein', i=1	‚test', i=2	Min
‚ganz', j=1	1•61	2•80	61
‚anders', j=2	2•103	1•94	94
Summe			155

Tabelle 3 - Satzabstand A-C

Anhand der Beispieltabellen lässt sich erkennen, dass Satz A und B einen geringeren Abstand ergeben als A und C. ♦

Weil die Relevanz nicht symmetrisch bezüglich der Parameter A und B ist, kann man bei Bedarf die Abstandsfunktion noch symmetrisieren mit

$$d_{satz}^{symm}(A,B)=\frac{d_{satz}(A,B)+d_{satz}(B,A)}{2}$$

Gleichung 4 - Satzdistanzsymmetrisierung

und verwendet die ebenfalls symmetrisierte Wortdistanzfunktion

$$d_{wort}^{symm}(a,b)=\frac{d_{wort}(a,b)+d_{wort}(b,a)}{2}$$

Gleichung 5 - Wortdistanzsymmetrisierung

Um eine Umrechnung in einen Relevanzwert für den Benutzer zu erreichen, der weniger an einer Metrik interessiert ist, invertiert und skaliert man den Wert mit

$$P(A,B)=\frac{1}{1+10d_{satz}(A,B)}$$

Gleichung 6 - Pseudowahrscheinlichkeit

und erhält eine Pseudowahrscheinlichkeit für die Ähnlichkeit der Beiden Sätze A und B.

4.3 Datenflussmodell

Um die Funktionalität der Spracheingabe und den zusätzlich notwendigen Funktionen einfach und fehlerfrei zu implementieren, wurde ein Datenflussmodell erstellt.

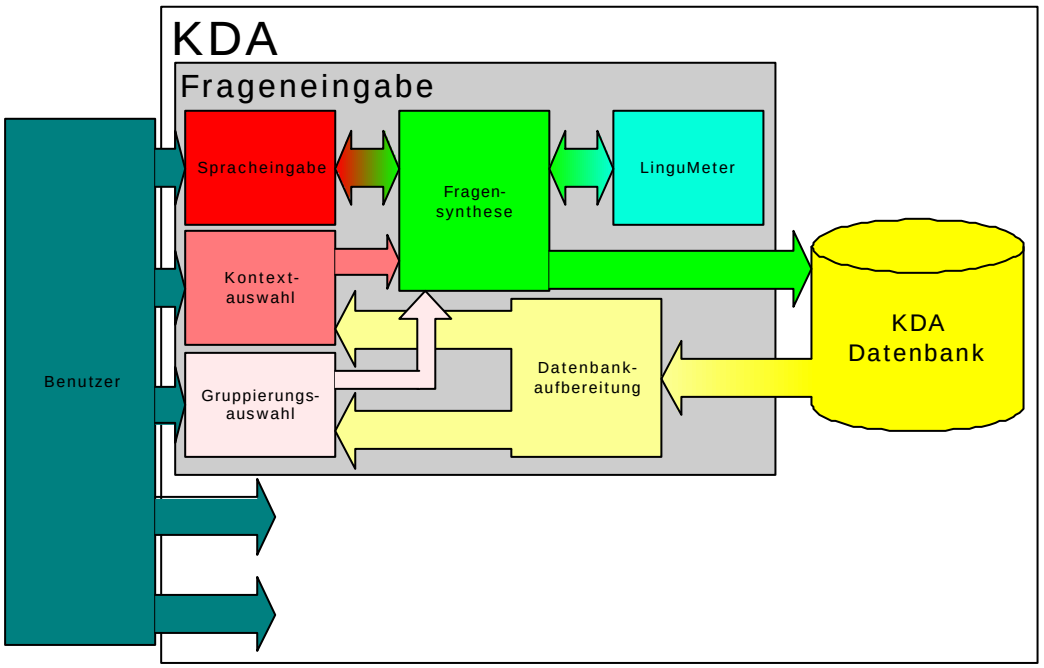

Abbildung 5 - Datenflussmodell Frageneingabe

Das Modell zeigt die Eingriffe des Benutzers in den KDA, welcher die Frageneingabe sowie weitere Elemente beinhaltet. Die Frageneingabe benötigt Informationen aus der KDA-Datenbank, noch bevor überhaupt eine Frage eingegeben werden kann, weil die gültigen Begriffe und deren taxonomischer Zusammenhang der Frageneingabe bekannt sein müssen. Das LinguMeter kommuniziert dabei ausschließlich mit der Fragensynthese um den Prozess der Fragemuster- und Argumentauswahl zu unterstützen. Der Benutzer hat direkten Zugriff auf die Spracheingabe, die Kontext- und Gruppierungsauswahl. Diese Eingabeelemente kommunizieren ebenfalls mit der Fragensynthese um die Benutzerwünsche entsprechend berücksichtigen zu können.

Die anderen Elemente bleiben hier unberücksichtigt, weil sie nicht im direktem Zusammenhang mit der Frageneingabe stehen. Obwohl es zum Teil vorteilhafter wäre, ein Klassenmodell zur Kommunikation zwischen Frageneingabe und KDA-Modulen zu erstellen, geschieht dies bislang über die KDA-Datenbank. In der Datenbank befinden sich die für die Fragen zulässigen Attribute, die Taxonomien für die Domänenbegriffe und weitere administrative Daten, welche vom KDA zur Laufzeit benötigt werden. Durch den Austausch dieser Datenbank ist es auch möglich, den KDA schnell auf eine andere Domäne mit anderen Taxonomien und Datenbeständen umzurüsten.

4.4 Benutzeroberfläche

Für den sichtbaren Teil der natürlichsprachlichen Frageschnittstelle, wurden verschiedene Designaspekte berücksichtigt. Ein Aspekt ist die schnelle Zugänglichkeit zu häufig verwendeten Informationen. Diese sollten nicht erst durch präzise Maussteuerung und durch drücken mehrerer Tasten erreichbar, sondern direkt, groß und deutlich im Fenster zu sehen sein. Eine gewisse Verteilung der Wichtigkeit von Informationen von links oben nach rechts unten sollte ebenfalls eingehalten werden wie auch eine klare Gliederung der Steuerelemente in geeignete semantische Gruppen. Diese und noch weitere ergonomische Gesichtspunkte ermöglichen eine einfach bedienbare Schnittstelle. Ein weiterer Designaspekt ist auch eine Oberfläche, welche mit möglichst wenig funktionalen Elementen auskommt und trotzdem die Vielfalt der möglichen Fragen nicht einschränkt. Im Folgenden werden die Schritte beschrieben, mit welchen eine KD-Frage konfiguriert wird.

4.4.1 Das Formular der Frageneingabe

Ein neues Frageneingabe-Formular enthält insbesondere im oberen Teil das Texteingabefeld zur Eingabe der Frage. Zusätzlich können im Feld „Vorschläge Eigenschaften“, welches ein Listenfeld enthält, Vorschläge für Attribute eingesehen werden, welche für die Formulierung der Frage verwendet werden können. Durch einen Doppelklick mit dem Mausknopf auf eines dieser Attribute wird dieses an der Aktuellen Eingabeposition im Texteingabefeld für die Frage eingefügt. Gruppenbildung und Kontext der Frage sind noch abgeschaltet. Meist werden diese zuletzt festgelegt, es ist jedoch auch möglich die Gruppenbildung und den Fragenkontext noch vor der Eingabe der Frage zu definieren.

Abbildung 6 - Neues Frageneingabeformular

4.4.2 Die Eingabe der Frage

Nach der Eingabe der Frage wird diese mit der Eingabetaste bestätigt. Dies aktiviert den Algorithmus zur Sprachanalyse. Je nach Geschwindigkeit der verwendeten Arbeitsstation erscheint spätestens nach einigen Sekunden die Liste der erkannten Fragen in absteigender Relevanz sortiert im Listenfeld in der Mitte der Frageneingabe.

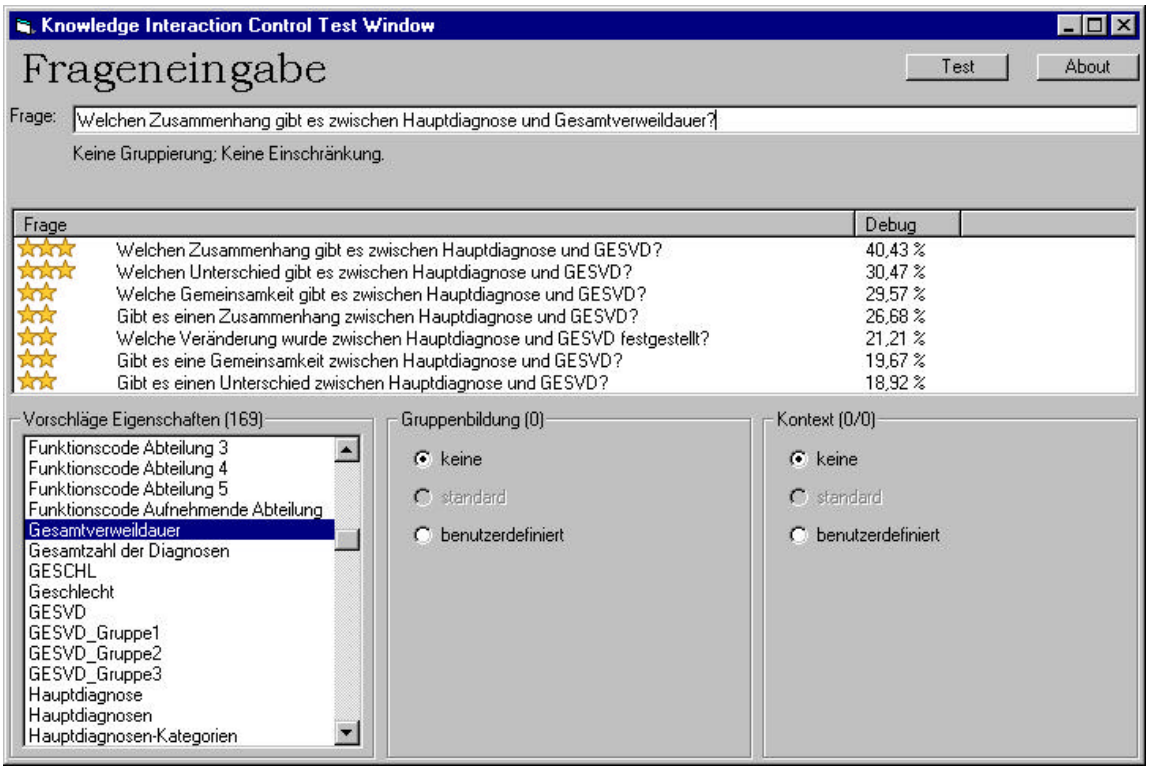

Abbildung 7 - Frageneingabeformular nach der Eingabe

4.4.3 Das Festlegen der Gruppenbildung

Möchte man seine Frage nicht über die Gesamtheit der verfügbaren Daten beantwortet haben, sondern nach bestimmten Attributen gruppiert haben, so kann man mit dem Feld „Gruppenbildung“ diesen Effekt erreichen. Es ist auch eine Gruppierung nach mehreren Attributen möglich. Wird die Gruppierung verwendet, wird die Frage für jeden Attributwert, welche in dem Gruppierungsattribut enthalten ist getrennt beantwortet.

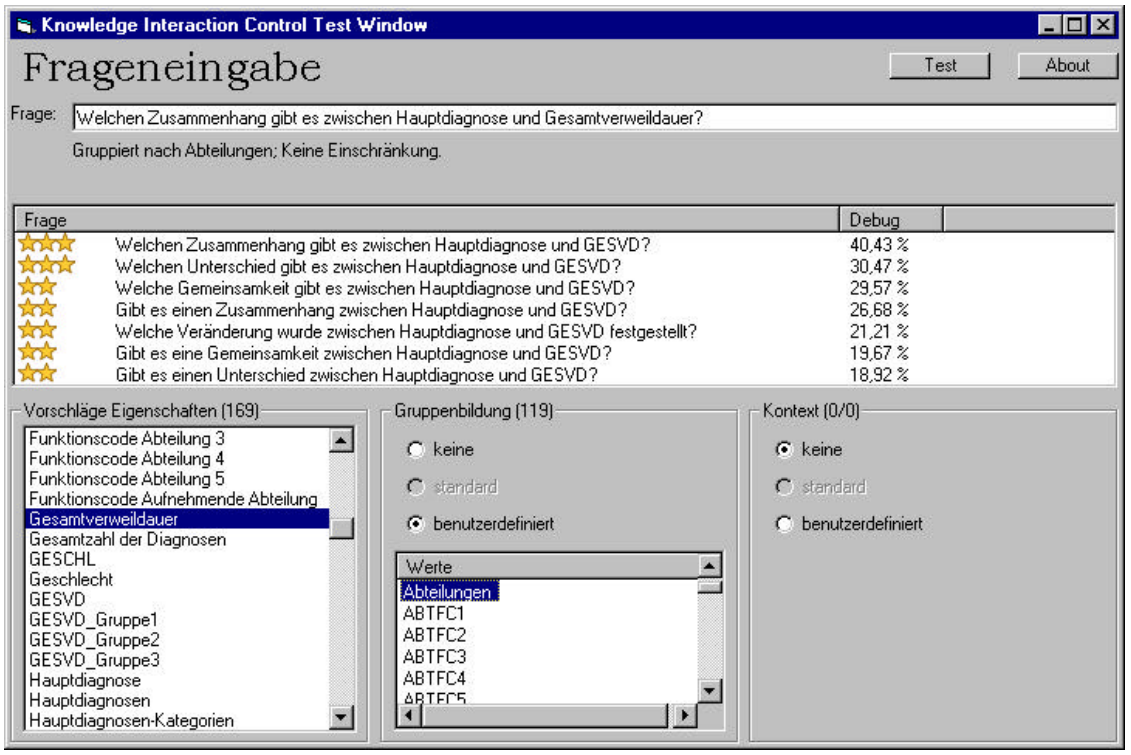

Abbildung 8 - Frageneingabeformular nach der Gruppenbildung

4.4.4 Die Auswahl einer Datensicht

Möchte man die Frage nicht auf den gesamten Datenbestand stellen, sondern nur für Teilmengen der Gesamtdaten beantwortet haben, so kann man das mit dem Fragekontext einstellen. Der Fragekontext verhält sich ähnlich wie die Gruppierung, nur dass die Selektion über die Auswahl einer Attribut-Attributwert-Kombination erfolgt.

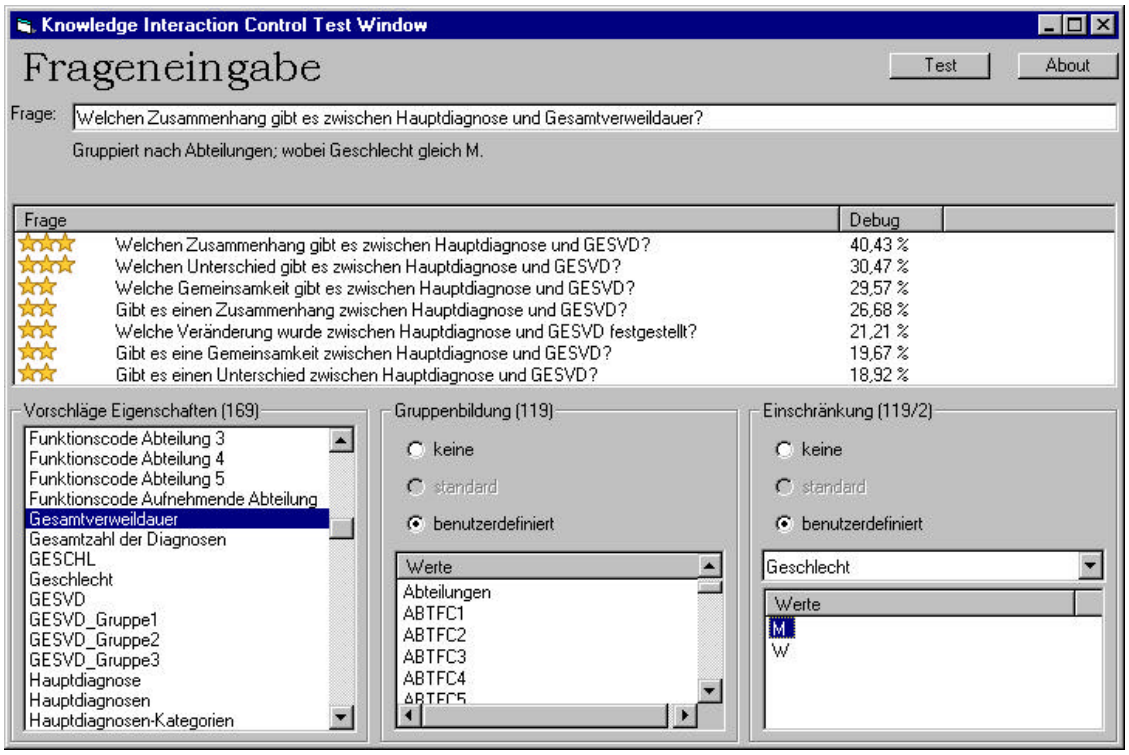

Abbildung 9 - Frageneingabeformular nach der Einschränkung

4.4.5 Formalisierte Ablage der Fragen

Wurde die Frage vom Benutzer eingegeben und als korrekt klassifiziert bestätigt, beziehungsweise vom Benutzer bis zur Bestätigung korrigiert, wird die Fragen in den KDA zur weiteren Verarbeitung weitergereicht. Dies geschieht über die KDA-Datenbank, welche eine Liste der bisher gestellten Fragen und deren Topologie enthält. Eine vom Benutzer als gültig gekennzeichnete Frage wird mit Hilfe von KDQL in die Datenbank eingetragen, wie in 3.1.1 erläutert wurde.

5 Implementierung der Frageneingabe und -analyse

Es lag nahe, die Algorithmen zusammen mit der Bedienerschnittstelle mittels eines ActiveX-Steuerelements zu implementieren. Da ein Teil der bestehenden Anwendung, der KDA-Client in Visual C++ aus MFC-Elementen konstruiert ist, erwies sich als Vorteil mit den bestehenden Komponenten über eine COM-Schnittstelle zu kommunizieren. Weiterhin besteht dadurch die Möglichkeit, die neue Benutzerschnittstelle auch in zukünftige Anwendungen einzubinden und damit wiederverwendbaren Code zu erzeugen.

5.1 LinguMetrie

Das LinguMeter implementiert die Komponente zur Bestimmung der Satzähnlichkeit. Im OLE-Container stellt die Komponente LinguMetrie die Schlüssel

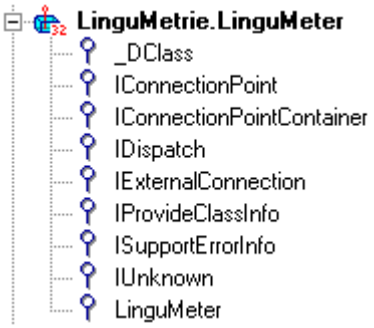

Abbildung 10 - Schlüssel LinguMeter

zur Verfügung.

Weil das LinguMeter keine graphischen Elemente enthält, ist eine einfache Implementierung ohne ActiveX-Zusatzkomponenten ausreichend. Die Implementierung erfolgt über die Schnittstelle

```
[
  uuid(CD95078D-1460-11D5-8BEC-FFFFFF000000),
  version(1.0),
  hidden,
  dual,
  nonextensible
]

dispinterface _LinguMeter {
    properties:
    methods:
      [id(0x60030001)]
      single WordDistance(
                  [in] BSTR s1,
                  [in] BSTR s2);
      [id(0x60030003)]
      single Distance(
                  [in, out] BSTR* s1,
                  [in, out] BSTR* s2);
};
```

Abbildung 11 - Interface Description Lanquage des LinguMeter

und stellt somit zwei Methoden zur Verfügung

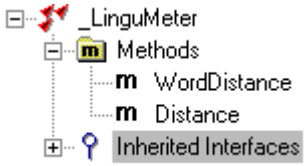

Abbildung 12 - Methoden LinguMeter

mit welchen es möglich ist, den Abstand zwischen zwei Wörtern und zwischen zwei ganzen Sätzen zu berechnen. Die beiden Methoden sind symmetrisch implementiert und lassen sich als .dll-Datei unter Beachtung der Urheberrechte in jedes Programm einbinden.

5.2 Knowledge Interaction Control

Das Knowledge Interaction Control oder kurz KIControl ist ein ActiveX-Steuerelement, welches auf COM basiert. Dieses implementiert einige zusätzliche Schnittstellen, welche eine grafische Darstellung innerhalb der standardisierten Windows-Umgebung ermöglicht.

Das KIControl stellt das eigentliche Interface für den Domänen-Experten dar, um seine Fragen an das Data-Mining-System zu stellen. Die Implementierung erfolgt inklusive Zugriff auf grafische Schaltelemente und OLE-Container-Methoden, welche eine Kommunikation zwischen KDA- und KIControl Komponente ermöglichen. Mit den Schlüsseln

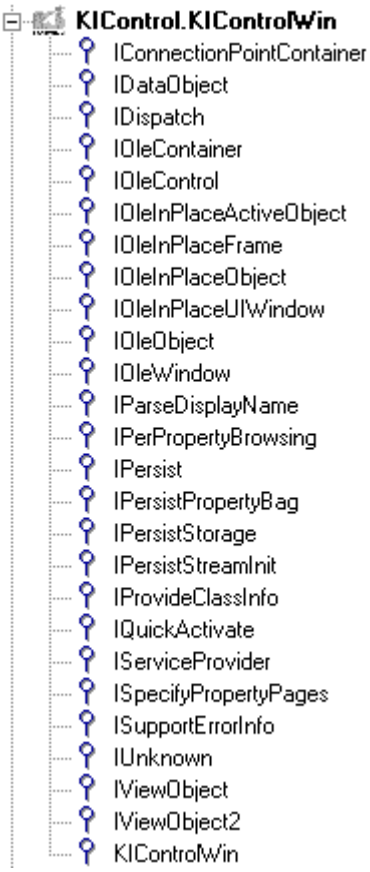

Abbildung 13 - Schlüssel KIControl

ist eine ActiveX-Steuerung sowie eine Kommunikation mit anderen Komponenten möglich, wie der Zusatz „Control“ im Namen der Komponente bereits suggeriert. Insbesondere der Schlüssel KIControlWin enthält die speziellen Kommunikationsmethoden für eine das Data-Mining-System durch die in Abbildung 14 - Interface Description Language des KIControl gezeigte Schnittstelle.

```
[
  uuid(CD95078F-1460-11D5-8BEC-FFFFFF000000),
  version(1.0),
  hidden,
  dual,
  nonextensible
]
dispinterface _KIControlWin {
    properties:
    methods:
};
```

Abbildung 14 - Interface Description Language des KIControl

und stellt damit die Methoden

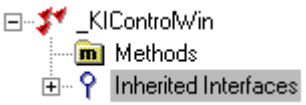

Abbildung 15 - Methoden KIControl

zur Verfügung.

6 Evaluierung

6.1 Sprachvermessung

Obwohl nicht gerade besonders viele Algorithmen für die Sprachvermessung verfügbar sind und deshalb die Implementierung des LinguMeter ein nützliches Werkzeug für das Data-Mining darstellt, bestehen weiterhin einige Probleme, welche zu einer Fehlklassifikation von Fragesätzen führen können. Zur näheren Bestimmung der Probleme bot es sich an, eine Frageliste zur Klassifikation durch das System zu erstellen und jeweils dazu passende manuelle Klassifikationen vorzunehmen. Damit ist es möglich, Veränderungen am Algorithmus vorzunehmen und mit geringem Aufwand Protokoll über das Leistungsverhalten der Sprachvermessung zu führen.

Probleme ergaben sich insbesondere bei der Erkennung der dynamischen Elemente der Fragemuster. Setzte man z.B. bei drei Frageparametern alle Kombinationen von nur jeweils zehn Elementen ein, so ergäben sich bereits tausend Mustervergleiche bei quadratischer Komplexität der Sprachvermessung. Hätte man in einer größeren Version des KDA noch fünfzig oder mehr Fragen zur Verfügung mit jeweils unterschiedlicher Parameterzahl, so wäre der Aufwand zur Abstandsberechnung auf einem gewöhnlichen Rechner nicht mehr akzeptabel, die Berechnung würde bis zu mehrere Stunden in Anspruch nehmen.

Bisher wurde versucht, das Problem durch „hartes“ Erkennen der dynamischen Parameter und anschließendem „weichem“ Erkennen der Gesamtfrage zu lösen. Eine Möglichkeit für eine Zwischenlösung wäre ein sukzessives Erkennen der Frageparameter statt diese simultan erfassen zu wollen. Während für die simultane Erfassung die Komplexität der kombinatorischen Satzvielfalt exponentiell zur Anzahl der Frageparameter steigt, blieb sie bei der sukzessiven Erfassung im linearen Bereich. Der Trick liegt dabei darin, den Mustervergleich nur bis einschließlich des ersten Parameters durchzuführen und den ersten Parameter anschließend aufgrund minimalen Abstands des Satzrudiments zur Fragestellung des Benutzers festzulegen. Man verfährt dann mit dem zweiten Parameter analog, setzt aber den ersten Parameter mit minimalem Abstand bereits ein, um die Verschiebung zu minimieren.

Die Kehrseite der Medaille bei dieser Form der Parametererkennung liegt darin, dass bei weicher Parametererkennung nicht nur das Argument in seinem Inhalt weich erkannt wird, sondern auch die Position des Arguments aufgeweicht wird. Fehlen nun dem eigentlichen Fragesatz Wörter im Vergleich zu dem Mustersatz mit dem verglichen werden soll, kann es leicht passieren, dass die Parameter an die falsche Position konvergieren. Im schlimmsten Fall konnte sogar beobachtet werden, dass Frageparameter schlicht vertauscht wurden. Dies ist fatal, da von jeder Frageart nur jeweils eine Version dem Benutzer vorgestellt wird und somit der Benutzer keine Möglichkeit mehr hat, die richtige Frage auszusuchen außer durch eine Neueingabe der Frage. Wegen dieser Schwachstellen wird vorerst die harte Erkennung der Frageparameter verwendet, bis eine Lösung der Problematik gefunden wurde.

Mittels des Klassifikationsfensters konnte bei jedem verändertem Parameter nun die Wirkung auf das Leistungsverhalten des Algorithmus geprüft werden. Die Ausgabe erfolgte dabei als eine Liste mit mehreren Spalten, wie in „Abbildung 16 - Klassifikationstest der Spracherkennung“ zu sehen ist.

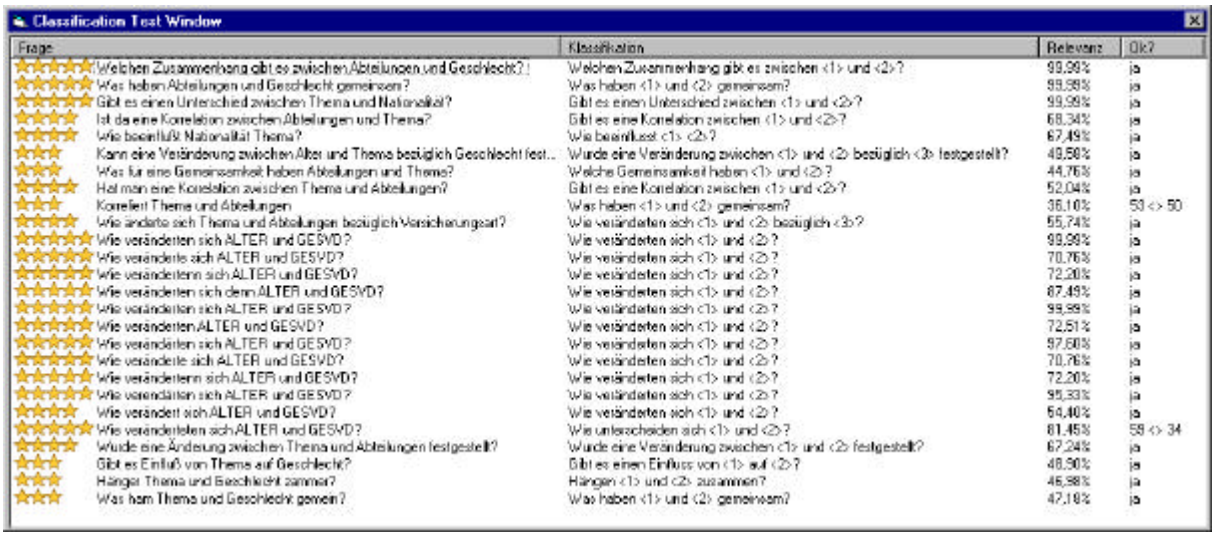

Abbildung 16 - Klassifikationstest der Spracherkennung

In der linken Spalte „Frage“ ist die überprüfte Frage zu sehen, die ersatzweise statt einer Benutzereingabe angewandt wurde. Die Klassifikationskonfidenz wird ebenfalls wie in der Benutzereingabe mit abgebildet. In der Spalte „Klassifikation“ ist die eingegebene Frage aus der Sichtweise der Sprachverarbeitung zu sehen. Nach der normalen Analysestufe wurde noch eine Synthesestufe nachgeschaltet, um die Frage wieder lesbar zu machen. Die Parametrisierung wird durch nummerierte Parameter „<n>“ mit $n = 1,2,\ldots$ angezeigt. Dabei befindet sich noch ein weiteres Feld „Relevanz“, welches die Ähnlichkeit des geprüften Satzes mit dem erkannten als Zahlenwert anzeigt, um während der Implementationsphase genauere Informationen einsehen zu können. Als letztes befindet sich noch die Spalte „Ok?“ in der Tabelle, welche anzeigt, ob die Klassifikation erfolgreich verlaufen ist bzw. wenn nicht zeigt diese Spalte die Primärschlüssel aus der Tabelle der zu klassifizierenden Fragen an. Weil die genaue Form der Fehlklassifikation meist keine weitere Rolle spielt, reicht die Anzeige der Primärschlüssel aus. Möchte man doch nachsehen, kann man dies immer noch in der entsprechenden Datenbanktabelle machen.

Durch diesen Evaluierungsmechanismus gelang es, die Zuverlässigkeit der Frageerkennung von etwa 70% korrekt klassifizierten Fragen auf über 90% korrekt klassifizierte Fragen zu steigern.

6.2 Praktische Ergebnisse

Es erweist sich als nützlich, die Parameter der formalisierten Data-Mining-Anfragen nicht explizit eingeben zu müssen. Da bisher noch keine vollständige Parametrisierung möglich ist, besonders in Bezug auf Fragekontext und Gruppierung, lässt die natürlichsprachliche Eingabe Wünsche offen.

7 Zusammenfassung, Diskussion und Ausblick

Sprachverarbeitung ist ein Thema, welches in seiner Entwicklung nicht nur eines der komplexesten ist, sondern auch noch sehr in den Kinderschuhen steckt. Wie man z.B. an IBM Via Voice oder anderen Spracherkennungssystemen für die gesprochene Sprache erkennt, besteht ein hoher Bedarf an anderen, vielleicht dem Menschen seiner Natur nach naheliegenderen Schnittstellen als Tastatur und Maus. Systeme zur Verarbeitung der gesprochenen Sprache werden sich kaum etablieren können, ohne dass eine Weiterverarbeitung der Syntax, Semantik und schließlich auch der Hermeneutik stattfinden wird.
Durch die Implementierung der natürlichsprachlichen Schnittstelle und der Verbesserung des Wissensstandes auf anderem Gebieten des Data-Mining haben sich weitere Möglichkeiten zur Weiterentwicklung des KDA aufgetan. Interessant ist insbesondere die Implementierung eines prozessorientierten Data-Mining-Modells, welches eine natürlichsprachliche Schnittstelle verfügt. Auch in Bezug auf die Implementierungsmethoden ist noch eine Weiterentwicklung vorstellbar. Ein Konsolidierungsschritt der Datenbank und eine Anpassung der KDA-Grundstruktur an moderne, komponentenbasierte Programmierung steht zur Diskussion. Ein mögliches Modell könnte dabei wie in „Abbildung 17 - KDA V2 Datenflussmodell“ aussehen, welches als Kernstück die natürlichsprachliche Kommunikation während des Data Minings mit dem Domänenexperten ermöglicht und somit das eigentliche Data Mining weitgehend in die Hand des Domänenexperten legt.

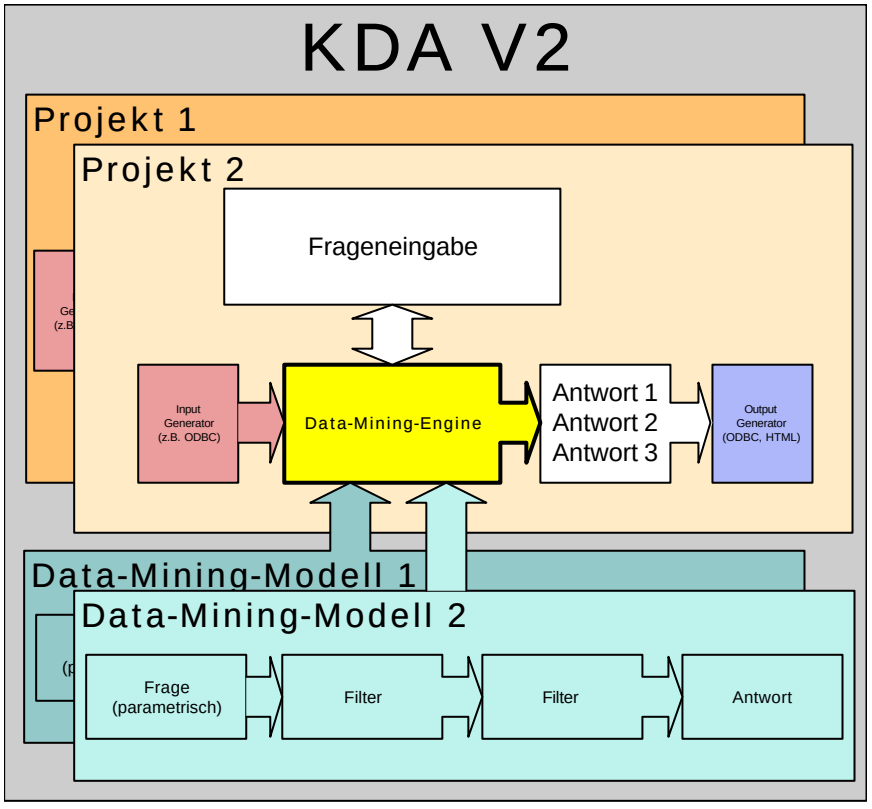

Abbildung 17 - KDA V2 Datenflussmodell

Das Herzstück des neuen KDA stellt die Data-Mining-Engine dar, welche in „Abbildung 18 - Data-Mining-Engine“. Die Data-Mining-Engine wird über den Input-Generator mit standardisierten Daten versorgt, welche z.B. aus ODBC-Quellen, Access-Files, Text-Dateien oder durch andere Module, die auch später hinzugefügt werden können geliefert werden. Der Input-Generator liefert neben den eigentlichen Daten noch die zum Data Mining notwendigen Metadaten, wie z.B. Spaltennamen, Spaltenzahlen, Datentypen und domänenspezifische Daten. Die Data-Mining-Engine versorgt angeschlossene Benutzerinteraktive Module wie die natürlichsprachliche Frageneingabe mit Informationen über die zur Beantwortung zur Verfügung stehenden Fragen, den Metadaten aus der Datenbank, um bestimmte Attribute zur Verfügung zu stellen. Vom Methoden-Modeller werden die eigentlichen Fragen mit den da-

zugehörigen Data-Mining-Methoden und einer Parameterbeschreibung geliefert. Aus den vom Benutzer gestellten Fragen generiert die Data-Mining-Engine durch Konkretisierung die notwendigen Fragebäume mit den konkreten Fragen und speichert zusätzlich die Topologie dieser Bäume, welche dann zur Beantwortung der abstrakten Fragen während der Antwortabstraktion herangezogen werden. Nach der Antwortabstraktion gibt die Data-Mining-Engine die beantworteten Data-Mining-Aufrufe nach außen um eine Weiterverarbeitung durch den Output-Generator zu ermöglichen, welcher Module wie HTML-Generator, ODBC-Speicherung, XML-Speicherung implementiert.

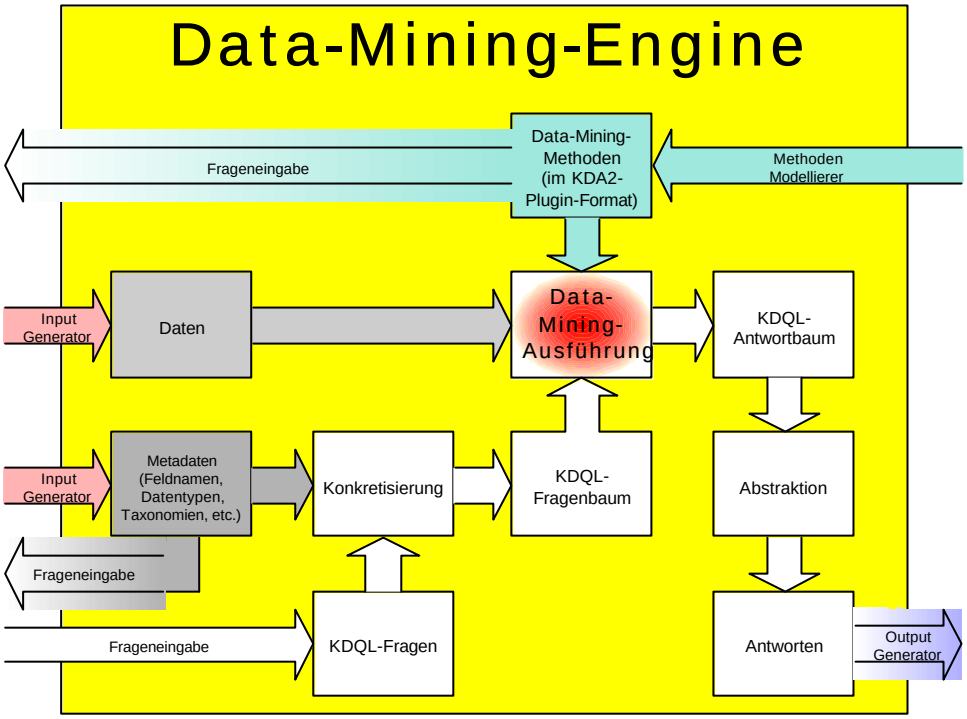

Abbildung 18 - Data-Mining-Engine

Dieser Modellierung kommen ganz natürlich die Erkenntnisse der Implementierung der natürlichsprachlichen Schnittstelle zugute, indem sich die Strukturierung der möglichen Fragen wesentlich verdeutlicht hat.

Literaturverzeichnis

[Als92] H. Alshawi. *The Core Language Engine*. MIT Press, Cambridge, Massachusetts, 1992.

[And93] I. Androutsopoulos, et al. *An Efficient and Portable Natural Language Query Interface for Relational Databases*. In P. W. Chung, G. Lovegrove and M. Ali, editors, Proceedings of the 6th International Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems, Edinburgh, U.K., pages 327-330. Gordon and Breach Publishers Inc., Langhorne, PA, U.S.A., Juni 1993.

[And94] I. Androutsopoulos et al.: *Natural Language Interfaces to Databases – An Introduction* ; Journal of Natural Language Engineering, Cambridge University Press; Research Paper no. 709, Department of Artificial Intelligence, University of Edinburgh, 1994.

[Aux86] P. Auxerre. *MASQUE Modular Answering System for Queries in English – Programmer's Manual*. Technical Report AIAI/SR/11, Artificial Intelligence Applications Institute, University of Edinburgh, März 1986.

[Bat86] M. Bates et al. *The IRUS transportable natural language database interface*. In L. Kerschberg, editor, *Expert Database Systems*, pages 617-630. Benjamin/Cummings, Menlo Park, CA., 1986.

[Ber97] Berson, A.; Smith S.: *Data Warehousing, Data Mining & OLAP*, McGraw-Hill, New York, 1997.

[Bin91] J.-L. Binot et al. *Natural Language Interfaces: A New Philosophy*. SunExpert Magazine, pages 67-73, Januar 1991.

[BNN89] *BNN Systems and Technologies*. BNN Parlance Interface Software – System Overview, 1989.

[Can92] Cannan, S.; Otten, G.: *SQL - The Standard Handbook*. McGraw-Hill, Berkshire, England, November 1992.

[Cod70] E. F. Codd. *A Relational Model for Large Shared Data Banks*. Communications of the ACM, 13(6):377-387, 1970.

[Cod74] E. F. Codd. *Seven Steps to RENDEZVOUS with the Casual User*. In J. Kimbie and K. Koffeman, editors, Data Base Management. North-Holland Publishers, 1974.

[Cop90] A. Copstake and K. Sparck Jones. *Natural Language Interfaces to Databases*. The Knowledge Engineering Review, 5(4):225-249, 1990.

[Dow81] D. R. Dowty et al. *Introduction to Montague Semantics*. D. Reidel Publishing Company, Dordrecht, Holland, 1981.

[Fay96] Usama M. Fayyad et al.: *Advances in Knowledge Discovery and Data Mining*; American Association for Artificial Intelligence, 1996; ISBN 0-262-56097-6.

[Gro83] B. J. Grosz. TEAM: *A Transportable Natural-Language Interface System*. In Proceedings of the 1st Conference on Applied Natural Language Processing, Santa Monica, California, pages 39-45, 1983.

[Gro87] B. J. Grosz et al. TEAM: *An Experiment in the Design of Transportable Natural-Language Interfaces*. Artificial Intelligence 32:173-243, 1987.

[Han96] J. Han, Y. Fu et al. *DMQL: A Data Mining Query Language for Relational Databases*, in: Proceedings of the SOGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, 1996.

[Har84] L. R. Harris. *Experience with INTELLECT: Artificial Intelligence Technology Transfer*. The AI Magazine, 5(2):43-50, 1984.

[Hau98] R. Hausser. *Häufigkeitsverteilung deutscher Morpheme*, LDV-Forum, Zeitschrift für Computerlinguistik und Sprachtechnologie 15/1, 6-28, 1998.

[Hen78] G. Hendrix et al. *Developing a Natural Language Interface to Complex Data*. ACM Transactions on Database Systems, 3(2):105-147, 1978.

[Hog01] Hogl, O. et al.: *On Supporting Medical Quality with Intelligent Data Mining*, in: Sprague, R. (Hrsg.): Proceedings of the Thirty-Fourth Annual Hawaii International Conference on System Sciences (HICSS-01), Maui, Hawaii, IEEE Press, Januar 2001.

[Hog98] O. Hogl: *Konzeption und Realisierung eines Data-Mining-Front-Ends zur Konkretisierung von Benutzerinteressen und eines Data-Mining-Back-Ends zur Abstraktion von Data-Mining-Ergebnissen*; Diplomarbeit Friedrich-Alexander-Universität Erlangen-Nürnberg, 1998.

[Jar85] M. Jarke et al. *A Field Evaluation of Natural Language for Data Retrieval*. IEEE Transactions on Software Engineering, SE-11(1):97-113, 1985.

[Joh85] T. Johnson. *Natural Language Computing: The Commercial Applications*. Ovum Ltd., London, 1985.

[Kle94] Klemettinen, M.; Mannila, H. et al.: *Finding Interesting Rules from Sets of Discovered Association Rules*, in: Adam, N.; Bhargava, B. et al. (Eds.): Third International Conference on Information and Knowledge Management, ACM Press, Gaithersburg, Maryland, 1994, pages 401-407.

[Mai83] M. G. Main und D. B. Benson. *Denotational Semantics for „Natural Language" Question Answering Programs*. Computational Linguistics, 9(1):11-21, Januar-März 1983.

[Mar86] P. Martin et al. *Transportability and Generality in a Natural-Language Interface System*. In B. J. Grosz, K. Sparck Jones and B. L. Webber, editors, Readings in Natural Language Processing, pages 585-593. Morgan Kaufmann Publishers, California, 1986.

[Ott92] N. Ott. *Aspects of the Automatic Generation of SQL Statements in a Natural Language Query Interface*. Information Systems, 17(2):147-159, 1992.

[Per88] C. R. Perrault und B. J. Grosz. *Natural Language Interfaces*. In H. E. Shrobe, editor, Exploring Artificial Intelligence, pages 133-172. Morgan Kaufmann Publishers Inc., San Mateo, California, 1988.

[Sch77] R. J. H. Scha. *Philips Question Answering System PHILIQA1*. In SIGART Newsletter, no.61. ACM, New York, February 1977.

[Sij93] W. Sijtsma and O. Zweekhorst. *Comparison and Review of Commercial Natural Languages Interfaces*. In F. M. G. de Jong and A. Nijholt, editors, Natural Language Interfaces, From Laboratory to Commercial and User Environments – Proceedings of the 5th Twente Workshop on Language Technology, Enschede, Twente University, NL, Juni 1993; MMC Pre-print no. 13, Institute for Language Technology and Artificial Intelligence (ITK), Tilburg University, NL.

[Sma83] D. W. Small und L. J. Weldon. *An Experimental Comparison of Natural and Structured Query Languages*. Human Factors, 25(3):253-263, 1983.

[Stü00] Stühlinger, W. et al. *Intelligent Data Mining for Medical Quality Management*, in: Lavrac, N. et al. (Hrsg.): The Fifth Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP-2000), Workshop Notes of the 14th European Conference on Artificial Intelligence (ECAI-2000), Berlin, August 2000.

[Wal78] D. L. Wanz. *An English Language Question Answering System for a Large Relational Database*. Communications of the ACM, 21(7):526-539, July 1978.

[War82] D. Warren, F. Pereira. *An Efficient Easily Adaptable System for Interpreting Natural Language Queries*. Computational Linguistics, 8(3-4):110-122, July-December 1982.

[Wit00] Ian H. Witten, Eibe Frank. *Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations*; Morgan Kaufmann Publishers, 2000; ISBN 1-55860-552-5.

[Woo72] W. A. Woods et al. *The Lunar Sciences Natural Language Information System* : Final Report. BBN Report 2378, Bolt Beranek and Newman Inc., Cambridge, Massachusetts, 1972.

Index