Suchfunktionalität in einem Webprojekt

Janet Bieneck, Robert Porscha, Enrico Treu

***Abstract*— Im Rahmen des Redesigns der tele-Task-Webplattform gilt es eine integrierte Suchfunktion für Wortvorkommen in Videoaufzeichnungen von Vorlesungen zu realisieren. Schwerpunkte der Umsetzung sind die Einbeziehung von Benutzerentscheidungen auf den Suchergebnissen für zukünftige Suchanfragen und die Darstellung der Suchergebnisse.**
Nach einer kurzen Vorstellung der allgemeinen Anforderungen und Rahmenbedingungen wird auf die Architektur des Systems eingegangen. Ausgehend von den Schwerpunkten folgt eine Einführung in das Gebiet der Support Vector Machines und die Vorstellung einer potenziellen Darstellungsmöglichkeit. Abschließend wird die Einbettung der Suchfunktion in das Gesamtprojekt angerissen.

***Index Terms*— Suchfunktionalität, Benutzerverhalten, Support Vector Machines (SVM), Darstellungsmöglichkeiten, Komponentenarchitektur**

I. Einleitung

Im Rahmen der Neuimplementierung und Erweiterung der tele-TASK-Webplattform soll eine Suche implementiert werden, die auf der Basis von Suchwörtern Zeitabschnitte in Vorlesungen, die als Video- bzw. Textdokumente vorliegen, findet und darstellt. Die darzustellenden Dokumente liegen zum Zeitpunkt der Ausarbeitung in Real[1]-Formaten (RealText,RealMedia) vor. Da diese Formate nicht in Klartextform sind, lassen sie eine direkte Suche anhand von Suchworten in den Dokumenten, ohne größeren Aufwand, nicht zu. Aus diesem Grund werden die Vorlesungen in eine Textform umgewandelt, die mittels textbasierter Suchverfahren durchsucht werden kann.

Die Erwartungen, die an eine Suchfunktionalität gestellt werden, sind stark beeinflusst von den Erfahrungen, die dem Benutzer vorliegen. Zum aktuellen Zeitpunkt entspricht dies in den meisten Fällen den Erfahrungen mit der Internet-Suchmaschine Google[2]. Diese bietet ein umfangreiches Angebot an Suchfunktionalität in verschiedenen Medien an. Im Gegensatz zu dem Umfang an Quellen und Medien, die durchsucht werden können, basiert die Darstellung der Suchergebnisse auf einer reinen Listenform, wodurch es bei einer großen Ergebnismenge zu Unübersichtlichkeit und Dopplungen kommen kann. Es wird zwar eine erweiterte Suche angeboten, allerdings wird auch hier nur eine einstufige Suche bzw. wortbasierte Suche durchgeführt. Das bedeutet, der Benutzer hat nur die Möglichkeit verfeinernde Suchbegriffe nachträglich einzugeben. Es werden keine Strukturierungsoptionen bereitgestellt bzw. nicht offensichtlich.

Ausgehend von den Erkenntnissen und Beobachtungen bei Google, sowie den spezifischen Anforderungen an die Suchlogik wurde die folgende Aufgabenstellung festgehalten.

A. Aufgabenstellung

Das Ziel des Projektes soll die Implementierung einer Suchfunktion für die tele-TASK-Webplattform sein, die es ermöglicht, innerhalb einer Textdatenbank Wortvorkommen zu finden. Die Textdatenbank, die hierfür verwendet wird, wurde mit Hilfe eines Texterkennungsprogrammes aus aufgezeichneten Vorlesungen (Video-Lectures) generiert. Während der Nutzung der Suche soll das Verhalten des Benutzers protokolliert werden, so dass dieses in nachfolgenden Suchanfragen berücksichtigt werden kann.
Schwerpunkte bei der Umsetzung sind die Suchlogik und deren Kapselung, damit diese später ersetzt werden kann, die Darstellung der Suchergebnisse und die Einbeziehung von Benutzerverhalten in die Auswahl bzw. Strukturierung der Suchergebnisse.

II. Rahmenbedingungen und Anforderungen

Ausgehend von der Aufgabenstellung und den gegebenen technischen Möglichkeiten, sind folgende Rahmenbedingungen und Anforderungen an die Umsetzung geknüpft.

[1] http://www.real.com
[2] http://www.google.com

A. Rahmenbedingungen

Die Rahmenbedingungen umfassen vor allem technische Gegebenheiten, die für die Erfüllung der Anforderungen, die im nächsten Abschnitt behandelt werden, notwendig sind. Diese können zwischen Gegebenheiten, die auf dem Webserver vorliegen müssen, und solchen, die auf der Seite des Klienten existieren, unterschieden werden. Nach einem kurzen Überblick, wird auf einzelne Technologien näher eingegangen.

Das Grundsystem basiert auf PHP 5.x und einer MySQL-Datenbank (5.1.x). Der Vorteil von PHP 5.x ist dessen verbesserte Umsetzung von Konzepten der objekt-orientierten Programmierung (Klassen/Objekte, Sichtbarkeiten, u.a.). Das bietet sowohl die Möglichkeit strukturierten Code zu entwickeln, der für spätere Entwickler leichter wiederzuverwenden ist, als auch die Definition von Klassen, die als Schnittstellen für andere Projekte bzw. Subprojekte dienen können. Außerdem wurden Funktionalitäten für Datenzugriffe auf der Basis von XML und Datenbanken (MySQL) in den Kern eingefügt bzw. verbessert.

Für die Darstellung der Zwischenergebnisse und den Videosequenzen lassen sich keine bzw. nur schwer technische Anforderungen stellen, da hier vor allem die Möglichkeiten des Klienten, sprich des benutzten Browsers entscheidend sind. Von Seiten der Anwendung wird deshalb auf der Basis der auf dem Klienten (Browser) vorhandenen Technologie entschieden, welche Darstellungsform gewählt wird.

Aktuell werden folgende Technologien berücksichtigt, von denen angenommen wird, dass sie zur Standardkonfiguration eines Browsers gehören:

- HTML
- Flash (Version 7+)
- JavaScript
- SVG (Scalable Vector Graphics)

Obwohl SVG zum aktuellen Zeitpunkt noch nicht in allen Browsern integriert ist, stellt es umfangreiche Möglichkeiten zur grafischen Darstellung zur Verfügung und bietet, im Gegensatz zu Flash, eine offene und standardisierte Form der Beschreibung von interaktiven und animierbaren Grafiken dar (siehe [1]). Wegen der Mächtigkeit von SVG gegenüber normalen binären Grafikformaten, wie PNG, GIF oder JPG, und dem verhältnismäßig geringen Platzbedarf gegenüber interaktiven Flashfilmen, wird es für die Darstellung in zukünftigen Systemen in Betracht gezogen. Aus dem oben genannten Grund, dass es noch nicht überall vollständig integriert ist, wird es in der ersten Implementierung nur eine untergeordnete Rolle spielen.

Ein besonderer Fall in dieser Liste stellt allerdings JavaScript dar. Für den Fall, dass das JavaScript auf dem Browser aktiviert ist, gibt es die Möglichkeit serverseitig auf Ajax aufzusetzen, um ein applikationstypischeres Reaktionsverhalten des Systems zu generieren. Allgemein ausgedrückt heißt das, das die Webseite sich ähnlich einer lokalen Applikation verhält, anstelle einer Webseite. Ein solches Verhalten zeigt sich z.B. durch die Möglichkeit ohne Neuladen der kompletten Seite auf Aktivierung von Optionen zu reagieren.
Ein Beispiel für ein solches Verhalten ist die Aktivierung einer Wortliste. Dabei wird zur Laufzeit ein Teil der Webseite vom Server nachgeladen und in die Webseite integriert, ohne das die entsprechende Webseite komplett neugeladen wird. (siehe [2])
Um Ajax auf dem Webserver verwenden zu können, muss eine zusätzliche Bibliothek vorliegen. Obwohl laut Aussagen in anderen Vorträgen innerhalb des Gesamtprojektes häufiger auf Ajax-Funktionalität zurückgegriffen wird, wird davon ausgegangen, dass Ajax nicht notwendiger Weise fest in den Webserver integriert ist. Aus diesem Grund ist eine Bibliothek, die sich unabhängig von sonstigen Webserverkomponenten nutzen lässt, vorzuziehen. Exemplarisch lässt sich hier ”xajax”[3] nennen. XAjax muss nicht zwingend fest installiert sein, sondern kann, ähnlich wie die nachfolgend vorgestellten Bibliotheken Smarty oder AdoDB, in einem Benutzerverzeichnis des Webservers angelegt werden und ist sofort einsetzbar. Weitere Informationen zu dem Thema sind in der Ausarbeitung über Ajax (siehe [2]) und http://www.xajaxproject.org zu finden.

[3] http://www.xajaxproject.org

Eine weitere Darstellungsmöglichkeit wurde nicht aufgezählt, da sie als Modul im Webserver integriert sein muss. Das GD[4]-Modul erlaubt es, direkt in einem PHP-Dokument Grafiken zu definieren anstatt diese in Form von Grafikdateien zu referenzieren. Da es sich hierbei um eine PHP-interne Funktionalität handelt, bzw. im Kontext von PHP und dem Webserver läuft, wurde sie, wie gesagt, nicht gesondert aufgeführt. Für den Fall, dass das Modul auf dem Webserver vorhanden ist, werden dessen Funktionen im Rahmen der HTML-Darstellung verwendet.

Auf die Kombination der einzelnen Technologien wird in den Anforderungen für die Darstellung (Abschnitt II-B.3) näher eingegangen.

Der Aufbau des Systems folgt der klassischen Aufteilung: Darstellung - Logik - Datenhaltung. Um die einzelnen Bereiche voneinander zu trennen wird auf bestehende Technologien aufgesetzt.
Für die Trennung von Darstellung und Logik wird Smarty, ein sogenanntes Template-System, eingesetzt. Die eigentliche Suchlogik wird von der Datenbank bzw. der Datenhaltung mittels einer Abstraktionsschicht, die durch die AdoDB-Bilbiothek realisiert wird, getrennt.

a) AdoDB: AdoDB ist eine Bibliothek für PHP und Python, die es erlaubt für verschiedene Datenbanktypen Anfragen in Form von SQL-Anweisungen zu senden. Dazu wird bei der Initialisierung der Datenbanktyp festgelegt.

```
include('/path/to/adodb.inc.php');
$DB = NewADOConnection('mysql');
```

Dieses Prinzip der Festlegung des Datenbanktyps wird von den Entwicklern bzw. den Vertreibern von AdoDB als "Driver"-Prinzip bezeichnet. Neben MySQL stehen weitere Datenbanktypen zur Verfügung, wie PostgreSQL, Firebird, Oracle, MS SQL.
Diese Bibliothek kann, aufgrund den darauf liegenden BSD-Lizenzbestimmungen, kostenfrei und ohne die Notwendigkeit der Zustimmung des Autors verwendet werden.

b) Smarty: Smarty ist ein sogenanntes Template-System für PHP. Ein Template ist eine Darstellungsmaske, in die Daten zur Laufzeit eingefügt werden können.

Beispiel eines Absatzes:

```
<p>
{$content}
</p>
```

Templates können aus kompletten Seiten, oder aus einzelnen Seitenfragmenten (wie das Beispiel) bestehen. Das System erlaubt es verschiedene Templates zu kombinieren. So kann das Beispieltemplate in ein Anderes eingebunden werden, welches für die komplette Seitendarstellung verantworklich ist.

"Einer der einzigartigen Aspekte von Smarty ist die Kompilierung der Templates. Smarty liest die Template-Dateien und generiert daraus neue PHP-Skripte; von da an werden nur noch diese Skripte verwendet."[3]

B. Anforderungen

In der Aufgabenstellung (siehe I-A) wurden bereits die allgemeinen Zielsetzungen an das System vorgestellt. An die einzelnen Teile sind jedoch spezielle Anforderungen geknüpft.

1) Datenverwaltung: An die Datenverwaltung sind keine speziellen Anforderungen gestellt. Um eine möglichst plattform-unabhängige Schnittstelle zu der verwendeten Datenbank zu gewährleisten, wird das AdoDB-System verwendet.

2) Suchlogik: Ziel der Suchlogik-Implementierung sollen sogenannte Support Vector Machines (SVMs) sein. Diese Art der Suchverfahren wird im Abschnitt IV genauer dargestellt. Um jedoch die Suche nicht auf eine Möglichkeit einzuschränken und um für Testzwecke eine vereinfachte Suchlogik zu implementieren, soll die Suchlogik gekapselt sein. In diesem Zusammenhang ist die Definition spezieller Anfrageobjekte notwendig, die nicht abhängig von der verwendeten Suchlogik sind. Die Komponente für die Suchlogik formt diese Anfrageobjekte in die für die Suchlogik entsprechende Form, z.B. in eine Datenbankabfrage, um. Durch den Einsatz von diesen, speziell für die Kapselung, konzipierten Anfrage- und Ergebnisobjekten wird

[4]Verfügbar unter http://www.boutell.com/gd/

außerdem noch das Risiko für Manipulation von Daten mittels SQL-Injection oder anderen Angriffsformen, die unter Umständen auf Zugriff auf sensible Komponenten des Netzwerkes ausgerichtet sind, minimiert. Die Umsetzung eines speziellen Sicherheitskonzepts ist allerdings nicht Ziel der Implementierung.

3) Darstellung: Die Darstellung der einzelnen Suchschritte erfolgt seitenbasiert. Im Gegensatz dazu stehen die Eintrittspunkte zur Suche. Es soll, angelehnt an das Modell von Google, sowohl ein Eingabefeld existieren, dass als Komponente in die Navigation oder an eine gesonderte Stelle innerhalb der allgemeinen Seitenstruktur integriert werden kann, als auch eine spezielle Suchseite, die verschiedene Konfigurationmöglichkeiten für die Suche anbietet.

Bei der Darstellung soll auf verschiedene Technologien, wie JavaScript, SVG, Flash zurückgegriffen werden können. Dies macht es notwendig diverse Flags für die einzelnen Technologien zu definieren. Für die aktuelle Implementierung wird jedoch nur JavaScript und Flash verwendet. Die Möglichkeit weitere Formate in der Darstellung zu verwenden, soll nicht durch spezialisierte Schnittstellen, z.B. das nur HTML-Formulare (bzw. PHP-Funktionen) für die Übertragung von Daten verwendet werden können, eingeschränkt oder gar verhindert werden.

Je nachdem welche Technologien vorhanden sind, wird eine entsprechende Darstellungsform ausgewählt. Stehen mehr als eine Technologie zur Verfügung (z.B. Flash und JavaScript) so soll der Benutzer über eine Option die Möglichkeit erhalten, eine Darstellungsform zu wählen.
Die Standardeinstellung ist hierbei die HTML-Darstellung in Verbindung mit JavaScript. Für den Fall, dass mehrere der Möglichkeiten zur Verfügung stehen, der Benutzer aber keine Auswahl trifft, werden, soweit es möglich und sinnvoll ist, die vorhandenen Technologien miteinander kombiniert. Entscheidungskriterium ist in diesem Zusammenhang, welche Technologien die größte Komfortabilität bezüglich der Benutzerinteraktion ermöglichen. Das heißt z.B. die Möglichkeit Tool-Tips (dynamisch einblendbare Informationsfelder) darzustellen, oder intuitive grafische Darstellungen verwenden zu können.
In Hinblick auf dieses Kriterium ergibt sich folgende Priorisierung:

- Flash
- JavaScript und SVG
- SVG
- JavaScript (Default)
- PHP-integrierte Grafikfunktionen

Der Einsatz von normalen HTML- bzw. PHP-Funktionen (z.B. die Einbindung von Image-Objekten) wird als selbstverständlich vorausgesetzt und deshalb nicht extra aufgeführt. Würde man es doch mit aufführen wollen, würde jeder Stichpunkt noch ein ”und HTML” enthalten.

Für die Trennung von Logik und Darstellung wird vorausgesetzt, dass ein Templatesystem vorhanden ist. Im aktuellen System wird sich hierbei auf Smarty gestützt.

Um das Benutzerverhalten zu protokollieren soll eine Suche mit mehreren Stufen implementiert werden.

Beispiel:
Nach der Eingabe der Suchworte, werden die gewünschten Vorlesungsreihen ausgewählt bzw. abgewählt, danach kann eine Einschränkung der darzustellenden Medien erfolgen, usw.

Abschnitt V geht im Detail auf die einzelnen Stufen der Suche, aus Sicht des Benutzers, ein.

III. Architekturübersicht

Um eine möglichst flexible Struktur zu erhalten, sind die einzelnen Elemente Darstellung, Suchlogik und Datenverwaltung voneinander getrennt. Dadurch sind einheitliche Schnittstellen notwendig. Abbildung 1 auf Seite 5 zeigt die allgemeinen Klassen bzw. Interfaces und deren Beziehungen.

A. Komponenten

1) Daten: Die Datenquelle, die als Grundlage der Suche verwendet wird, ist die bereits in der Einleitung erwähnte Wortdatenbank. Angelegt wurde diese Datenbank mit Hilfe eines Spracherkennungsprogramms, das von Stephan Repp, einem externen Doktoranden von Professor Doktor Meinel, im Rahmen seiner Promotion

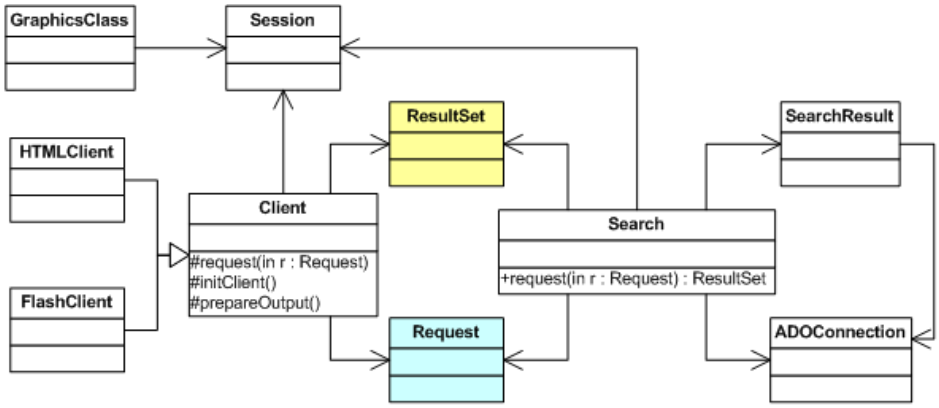

Fig. 1. Klassendiagramm

erstellt und an der Vorlesungsreihe "Technische Grundlagen des WWW"(SS 2005) getestet wurde. Sie enthält, auf verschiedene Tabellen verteilt, die gesprochenen Worte von Vorlesungen in Textform mit einem Zeitstempel versehen, Synonyme für typische Begriffe und Zeitintervalle, sogenannte Zeitcluster, in denen ein Begriff besonders gehäuft auftritt. Abbildung 2 (Seite 13) zeigt das Schema des entsprechenden Datenbankabschnitts. Der Zugriff auf die Datenbank erfolgt mittels der `ADOConnection`-Klasse. Sie verwaltet den Zugriff auf die jeweilige Datenbank. Somit lassen sich unterschiedliche Datenbanktypen einsetzen, ohne dass die übrigen Klassen davon beeinflusst werden. Um eine weitere Trennung zu erreichen, ist es denkbar eine zusätzliche Abstraktionsschicht einzusetzen, so dass die Klasse für die Suchlogik nicht zwingend eine Datenbankanfrage formulieren muss, sondern ein selbstdefiniertes Datenbankrequest-Objekt an den Datenbankverwalter schickt. Damit wäre man komplett unabhängig von der Speicherungsform der Daten, welche z.B. in XML-Form vorliegen können. Von dieser zusätzlichen Abstraktion wird aber vorerst Abstand genommen und nur die bestehende Datenbank als Datenquelle zugrunde gelegt.
Das `SearchResult` entspricht dem ResultSet, das aus der Datenbankabfrage generiert wurde.

2) Suchlogik: Wie bereits in den Anforderung an die Suchlogik erwähnt, soll die Implementierung des Suchalgorithmus nach außen nur über eine generische Schnittstelle sichtbar sein, sodass keine Abhängigkeiten bezüglich der Darstellung und eines spezifischen Suchalgorithmusses entstehen. Um diese Trennung von Client vom Suchalgorithmus zu gewährleisten, werden spezielle Request- und Resultobjekte definiert. Je nachdem welche Informationen zurückgeliefert werden sollen, werden spezielle Requestobjekte verwendet.

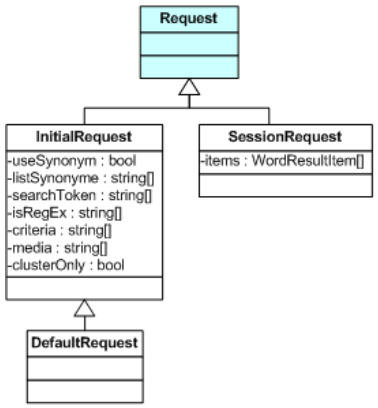

Fig. 3. Request-Klasse

Für Anfragen, die eine Abarbeitung des Suchalgorithmus erfordern, werden die sogenannten ”InitialRequests” gesendet. Für Anfragen, die bereits während einer Benutzer-Session verarbeitet wurden, und die dadurch bereits vom Session-Objekt zusammen mit dem entsprechenden Ergebnis bereits gespeichert wurden, können mit Hilfe des Session-Requests abgefragt werden. Dieser Anfragetyp dient zur Minimierung des Verarbeitungsaufwands für sich wiederholende Anfragen. Eine ”wiederholte Anfrage” ist eine Anfrage, die die gleichen Suchbegriffe in der gleichen Reihenfolge bzw. mit den

gleichen Suchoptionen enthält. Dadurch wird eine Verbesserung der Laufzeit erwartet. Nachteil dieses Verfahrens ist, dass während einer Session, bei sich wiederholenden Anfragen das bereits erfolgte Benutzerverhalten nicht berücksichtigt wird. Das bedeutet, dass die Ergebnisse, inlusive denen, die bei der vorherigen Anfrage als nicht-relevant gekennzeichnet wurden, in der gleichen Reihenfolge angezeigt werden, wie bei der vorherigen Anfrage. Falls nach oder während der Implementierung offensichtlich wird, dass diese Maßnahme keinen ausschlaggebenden Zeitvorteil gegenüber einer komplette Verarbeitung der Suchanfrage ergibt, wird diese Funktion aus dem System entfernt. Für die

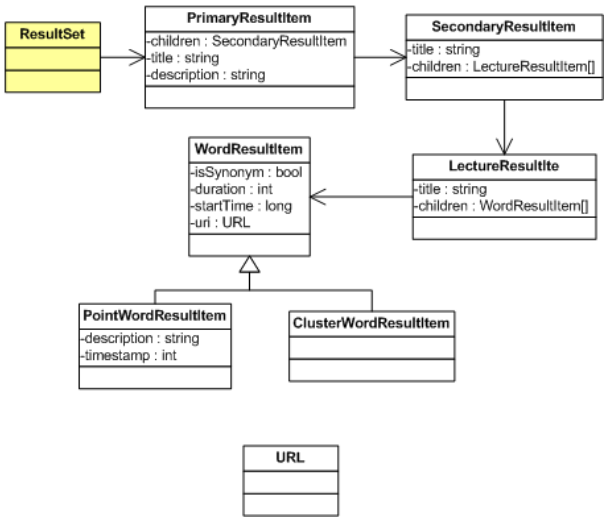

Fig. 4. Result-Klassen

Rückgabe der Ergebnisse ist ebenfalls eine Gruppe von Klassen notwendig. Die ResultSet-Klasse enthält eine Liste von Resultobjekten. Diese werden anhand der Sortierkriterien, sowie dem Typ des Treffers strukturiert. Auf die Sortierkriterien wird im Abschnitt V-B näher eingegangen. Die Unterscheidung zwischen Zeitpunkt-Treffer und Cluster-Treffer rührt aus dem Aufbau der Datenbank her. In dieser wird zwischen Einzelvorkommen eines Wortes (Zeitpunkt) und dem gehäuften Vorkommen innerhalb eines definierten Zeitraumes (Cluster) unterschieden. Der Zeitraum für ein Cluster wird derzeit von den Erstellern des Datenbankinhalts festgelegt.

3) Session: Die `Session`-Klasse stellt die Protokollierungsfunktionalität zur Verfügung. Für die spätere Einbindung vorheriger Sessiondaten in die Suchlogik, wird die `Search`-Klasse auf die `Session`-Klasse zugreifen. Welche Daten protokolliert werden und in welcher Form sie gespeichert werden hängt von den Daten ab, die der Suchalgorithmus benötigt.
Um an dieser Stelle so unabhängig wie möglich vom Suchalgorithmus zu sein, werden die einzelnen Request- und Resultobjekte gespeichert. Über eine längerfristige und somit persistente Speicherung der Daten wurde zum aktuellen Zeitpunkt noch nicht entschieden, da sich damit Fragen der Zuordnung zu Benutzern, sowie des Umfangs der gespeicherten Daten und deren Auswertungsmöglichkeiten ergaben.

4) Clients: Um verschiedene Client-basierte Technologien (Flash, JavaScript, etc.) nutzen zu können und neue Klientenklassen, ohne große Reimplementierung des übrigen Systems, einzubinden, basieren die Darstellungen für den Klienten auf der abstrakten `Client`-Klasse. Sie stellt die grundlegende Schnittstelle zum Absenden und Empfangen von Suchanfragen in Form von Request- und Resultobjekten dar.

IV. Suche

Die Grundidee hinter der Suche ist, das zu einem oder mehreren Suchbegriffen die entsprechenden Vorkommen innerhalb der gegebenen Veranstaltungstexte gefunden werden. Dabei gilt es jedoch außerdem für den Benutzer die relevanteren Ergebnisse hervorzuheben. Was als relevantes Ergebnis angesehen wird, ergibt sich dabei aus der Kombination von Worthäufungen und benutzerabhängigen Entscheidungen, die sich auf spätere Suchanfragen auswirken können.

Die Findung von Wortvorkommen innerhalb der Textquellen lässt sich mittels einer Datenbankabfrage realisieren. Dabei wird eine Unterscheidung getroffen zwischen Treffern, die zu einem bestimmten Zeitpunkt auftreten und Vorkommen innerhalb eines Clusters.
Diese Unterscheidung wird anhand einer Gewichtung erreicht. Hierbei werden Clustertreffer höher bewertet als Zeitpunktstreffer. Mögliche Überschneidungen von unterschiedlichen Treffertypen werden dergestalt berücksichtigt, dass Einzeltreffer, die innerhalb eines Clusterintervalls auftreten aus der Ergebnismenge entfernt werden. Insofern es sich bei dem Einzeltreffer um einen zusätzlichen Begriff handelt (z.B. ein

Synonym des eigentlichen Suchwortes, oder ein weiteres Suchwort) wird der Wert für das Cluster erhöht. Über die Skalierung und Bewertung der Ergebnisse kann zum aktuellen Zeitpunkt noch keine Aussage getroffen werden, da dies erst in der Implementierungsphase nochmals genauer untersucht wird.

A. Support Vector Machines

Wie bereits in den Anforderungen erwähnt, ist das Ziel bei der Implementierung der Suchlogik ein System, dass nicht nur alle Suchergebnisse anhand der Wortvorkommen ausgibt, sondern diese Anhand einer Art von Kontext strukturiert, so dass relevante Ergebnisse hervorgehoben werden. Dabei sollen vor allem Benutzereingaben - sowohl Suchworte, als auch Entscheidungen über die gegebenen Suchergebnisse - berücksichtigt werden. Eine Möglichkeit dies zu erreichen, ist der Einsatz von sogenannten lernfähigen Systemen. Zu dieser Kategorie gehören Support Vector Machines (SVMs). SVMs wurden in 1979 von V. Vapnik et al. entwickelt. Sie basieren auf dem ”structural risk minimization principle”[4] der statistischen Lerntheorie und sind fähig lineare Entscheidungsregeln $h(\vec{x}) = sign(\vec{w} \bullet \vec{x} + b)$, die durch einen Wichtungsvektor w und einem Grenzwert b beschrieben werden, zu erlernen. Für linear separierbare Trainingsbeispiele finden SVMs eine ”Hyperplane”[4] mit der maximalen Euklidischen Distanz zu den nähesten Trainingsbeispielen.
Trotz dieser theoretischen Definition soll an dieser Stelle jedoch keine mathematische Einführung in dieses Feld erfolgen, da dies den Rahmen dieser Ausarbeitung sprengen würde, sondern eher eine aufgabenbezogene Vorstellung der Grundprinzipien und Möglichkeiten von SVMs.

Da der Begriff Relevanz keinen allumfassenden Wert besitzt, sondern individuell für jede Person unterschiedlich ist, ist es leicht verständlich, dass die Festlegung relevanter Ergebnisse nicht anhand einer klassischen Implementierung möglich ist. Die Familie der lernfähigen Systeme bietet für diese Art von Problemen eine komfortable Lösung. Sie sind in der Lage aus den Entscheidungen des Benutzers zu ”lernen” und können dadurch bei zukünftigen Suchanfragen auf ihr Wissen zurückgreifen und die Ergebnisse entsprechend strukturieren.

Die für die Aufgabe der Suche interessanten Systeme sind sogenannte Klassifizierer (engl. *Classifier*). Sie arbeiten auf der Basis von Funktionen, die eine Datenmenge in Klassen unterteilen. Im konkreten Fall in relevante und irrelevante Daten. Dazu wird die Maschine anhand von Beispielen ”trainiert”. Ziel dieses Trainings ist, die Funktion, die meist auf statistischen Grundlagen beruht, soweit zu verallgemeinern, dass sie eine möglichst sinnvolle Klassifikation erreicht.
Um dies kurz an einem allgemeinen Beispiel zu verdeutlichen:
Zwei Brüder sind Botaniker. Beiden wird eine Pflanze gegeben, die sie anhand eines Beispielbaumes entweder als Baum oder als Nicht-Baum klassifizieren sollen. Der eine Bruder ist sehr penibel und zählt jedes einzelne Blatt und entscheidet sich gegen die Klassifizierung als Baum, da sich die Blattanzahl gegenüber dem Beispielbaum unterscheidet. Der andere Bruder ist etwas ungenauer und sieht nur dass die Pflanze grün ist und katgorisiert sie damit automatisch als Baum.

Anhand dieses kurzen Beispieles wird deutlich, dass eine Balance zwischen Genauigkeit und Fehlertoleranz gefunden werden muss.

1) Training: Bevor eine reale Aufgabe bearbeitet werden kann, müssen die zu bearbeitenden Objekte in eine für die Maschine verständliche Form gebracht werden. Dies bedeutet in Verbindung mit SVMs die Umformung in eine sogenannte *feature vector representation*. Die Attribute eines Objektes müssen in eine Vektorform überführt werden. Dabei muss darauf geachtet werden, dass die Form und Auswahl der Attribute für die Lernfunktion geeignet ist. In der Literatur findet man an einigen Stellen die Unterscheidung zwischen *Feature* und Attribut. Ein Objekt besitzt Attribute, die durch die Überführung *Features* werden. Bezogen auf die Aufgabenstellung bietet sich an, die Häufigkeit des Auftretens von Worten als Attribute zu verwenden. Um die Menge zu minimieren und damit den Prozess zu optimieren werden keine Worte wie ”und”, ”oder”, etc. aufgenommen, da diese sehr oft auftreten und den Text nicht oder nur indirekt charakterisieren.
Wichtig bei der Wahl der Attribute ist, dass sie sich ordnen lassen, also eine Ordinalskala angewendet

werden kann.
Mathematisch wird ein Objekt durch ein Paar (x, y) definiert, wobei x ein Vektor mit den Features (Daten) darstellt und $y \in \{+1, -1\}$ die Relevanz oder Nicht-Relevanz charakterisiert.
Das Training erfolgt dann mit Hilfe von Trainingsbeispielen, sogenannten *Samples*, wobei jedes Sample durch ein Paar (x, y) definiert ist. Außerdem können, je nach Aufgabenstellung, weitere Informationen im Sample enthalten sein. Zum Beispiel beim Training eines Buchstabenerkennungssystems wird der zu erkennende Buchstabe vorgegeben und dann erhält die Maschine die Daten über die Linienzüge dieses Buchstabens. Die Anzahl der Trainingsbeispiele ist nicht begrenzt, und ist abhängig von der gewünschten Qualität der Klassifikationsfunktion.
Ziel des Trainings ist die Erstellung einer Geraden, die den Vektorraum der Trainingsdaten dergestalt teilt, dass die relevanten Objekte auf der einen Seite liegen und die irrelevanten auf der Anderen. Diese Gerade muss zusätzlich das Kriterium erfüllen, dass der Abstand der am nächsten liegenden Objekte maximal ist. Das bedeutet, dass entlang der Geraden auf beiden Seiten ein Bereich liegen muss, in dem sich keine Objekte befinden (siehe Abbildung 5). Dabei werden die Objekte, die die Breite der Bereiche bestimmen, als Support-Vektoren, wodurch die Maschine ihren Namen erhalten hat. Normalerweise

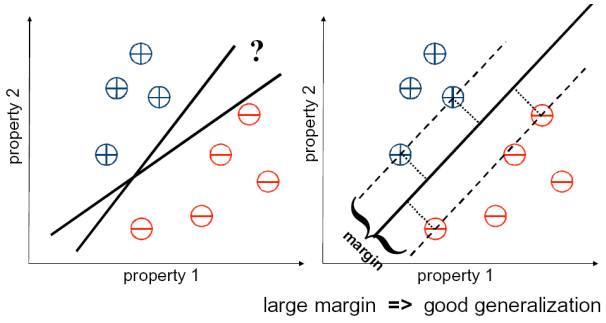

Fig. 5. Je größer der Abstand der Gerade zu den Support-Vektoren, desto besser die Generalisierung (Quelle:[5])

wird beim Training nur eine allgemeine Klassifizierung trainiert. Das heißt, unabhängig vom Benutzer, der eine Klassifikationsanfrage stellt, wendet die Maschine nur die eine trainierte Funktion an. Da im Falle der Suche die Relevanz der Ergebnisse stark vom Benutzer abhängig ist, bietet sich ein Training der Maschine anhand der Benutzereingaben während einer Session an. Dies hat zwar den Vorteil, dass es sehr individuelle Ergebnisse erzeugt, aber zieht auch diverse Probleme mit sich.

a) Das erste große Problem ist die Menge an Trainingseinheiten, die das System benötigt, um eine sinnvolle Klassifizierung durchführen zu können. Das Training einer SVM kann, abhängig vom Themenbereich, im Schnitt mehrere zehntausend Trainingseinheiten umfassen. Dies entspräche, nähme man eine Frequenz von ca. 5 Anfragen pro Session als Grundlage, etwa 2000-20000 Sessions bis die Maschine eine optimale Klassifizierung vornimmt. Dies macht es entweder notwendig, dass die Maschine unabhängig von den Benutzern auf ein gewisses Grundwissen trainiert wird, oder das Training, bezogen auf einen Benutzer, mit Hilfe von anderen Benutzereingaben angereichert wird. Beide Maßnahmen hätten zur direkten Folge, dass die Maschine keine komplett benutzerabhängige Klassifizierung durchführt, sondern nur eingeschränkt durch den Benutzer geprägt wird.
b) Das zweite Problem, das mit dem ersten zusammenhängt, ist der Initialzustand des Systems. Das heißt der Zustand in dem die Maschine bzw. das System ”ausgeliefert” wird. Ein lernendes System verfügt im Initialzustand über keine Informationen und gibt dementsprechend relevante, wie auch irrelevante Ergebnisse zurück. Dies stellt zwar im Allgemeinen kein Problem dar, da dies in etwa dem Verhalten einer Suchmaschine entsprechen würde, die nur auf Wortvorkommen achtet, widerspricht aber den funktionalen Anforderungen, dass nach benutzerabhängiger Relevanz geordnet werden soll.
c) Ein weiteres Problem stellt die Laufzeit dar. Führt man den Lernalgorithmus während der Suchanfragen aus, so kann dies zu Verzögerungen führen, die der Nutzer als unnötig oder unverständlich ansehen kann.
d) Ein allgemeines Problem, dass lernfähgie Systeme unter Umständen aufweisen können hängt mit der Frage der persistenten Speicherung von Trainingsergebnissen zusammen. Wie speichert man ”Lernerfolge” ab?

Im Zuge der Implementierung der Support Vector Machine werden diese Probleme untersucht und im Rahmen der Auswertung der Implementierung darauf eingegangen.

2) Klassifizierung: Die Möglichkeit des Trainings ist nur eine unterstützende Funktion, wenn auch Hauptcharakteristik eines lernfähigen Systems.

Die Hauptfunktion besteht darin eingegebene Daten bzw. Objekte mit Hilfe der trainierten Funktion zu klassifizieren. Dabei werden die durch das Training priorisierten Attribute verglichen und im Idealfall der richtigen ”Seite” zugeordnet. Das mathematische Modell ist hierfür um einiges komplexer als es die Erklärung vermuten lässt. Eine mögliche Grundlage der Zuordnung sind Formeln, mit dessen Hilfe der Normalvektor bezüglich der Trenngeraden und dem zu klassifizierenden Objekt ermittelt wird. Von diesem Vektor wird das Vorzeichen bzw. das Signum genommen. Ist das Signum -1 so ist das Objekt irrelevant, nimmt es den Wert +1 an, so ist das Objekt relevant.
An dieser Stelle tritt ein großer Vorteil von SVMs zu Tage. Bei lernfähigen Systemen gibt es einige Hauptproblematiken. Eine davon ist das sogenannte *Overfitting*. Mit Overfitting wird das Phänomen bezeichnet, dass das System bzw. die Maschine zwar immer größere Erfolge bei der Klassifizierung der Trainingsdaten erzielt, aber bei Testobjekten, die nicht den Kriterien der Trainingsdaten entsprechen, mit zunehmender Komplexität des Objektes versagt. Versagen heißt in diesem Zusammenhang, dass die Fehlerquote sehr hoch ist oder steigt. SVMs sind

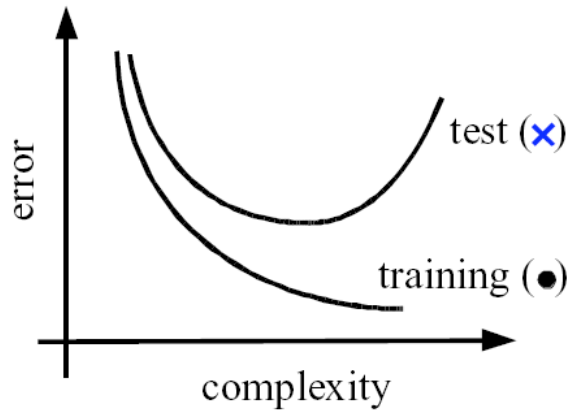

Fig. 6. Overfitting-Problem anhand eines Diagramms (Quelle:[5])

weniger anfällig gegenüber Overfitting.

Schwierigkeiten bei der Klassifikation: Im Gegensatz zu den abstrakten Beispielen, wie sie häufig in der Literatur verwendet werden, besteht hier die Eingabe nicht aus einem zu klassifizierenden Objekt, sondern aus Daten, die auf die Objekte angewendet werden und dadurch eine Klassifikation forcieren. Das bedeutet, dass nicht geprüft wird, ob die Eingabe relevant ist, sondern es wird davon ausgegangen, dass die Eingabe ein Relevanzkriterium darstellt. Eine weitere Schwierigkeit ergibt sich durch die Kombination von Wortvorkommen und Vorlesungen. Theoretisch ließe sich jedes Wortvorkommen klassifizieren, da dies im letzten Schritt die Ergebnisse sind, die der Benutzer als relevant oder nicht relevant charakterisiert. Alternativ lassen sich aber auch komplette Vorlesungen bewerten. Die Frage, die sich hierbei stellt, ist, was ist für den Benutzer ausschlaggebender bzw. inwieweit beeinflusst die Relevanz einer Vorlesung die Relevanz eines Wortvorkommens innerhalb der Vorlesung oder umgekehrt.

V. Darstellung

A. Suchmaske

Ausgehend von dem bereits erwähnten Suchverhalten mit Hilfe von Google (siehe Einleitung), stehen zwei Suchmasken zur Verfügung. Die erste ist eine Standardsuche mittels eines einfachen Eingabefeldes, das in jede beliebige Seite eingebunden werden kann. Die zweite Suchmaske ist eine Art erweiterte Suche in der z.B. folgende Suchkriterien eingestellt werden können:

Art der Wortverknüpfungen (UND, ODER): Parallel zur Möglichkeit innerhalb der Such-Zeichenkette Kombinationssymbole zu setzen (z.B. +,-) können mit dieser Option alle Suchbegriffe UND- oder ODER-verknüpft werden

optionale Synonymsuche: Die Zuhilfenahme von Synonymen zu den eingegebenen Suchbegriffen muss explizit aktiviert werden, bevor anhand eine Liste von möglichen Synonymen angelegt oder editiert werden kann. An dieser Stelle wird für die Implementierung vorausgesetzt, das JavaScript aktiviert ist, da hierfür ein Nachladen der Liste zur angezeigten Webseite notwendig ist. Für den Fall das JavaScript nicht verfügbar ist, wird die komplette Seite neu geladen, wobei die bereits gesetzten Optionen übernommen werden.

optionale reguläre Ausdrücke: Die Möglichkeit reguläre Ausdrücke zu verwenden muss explizit festgelegt werden. Ansonsten werden Symbole, die als Platzhalter in regulären Ausdrücken gelten, als zu suchende Zeichen angesehen und nicht interpretiert.

Eingrenzung der Medientypen: Der Benutzer soll die Wahl haben, welche Medientypen (z.B. Video oder nur Audio) er ausgegeben bekommen haben

will. Da diese Funktionalität bereits von einer anderen Gruppe übernommen wird, bleibt es derzeit offen, ob sie an dieser Stelle implementiert wird. Eine weitere Schwierigkeit, die sich bei Planungsgesprächen ergeben hat, ist, ob der Zugriff auf diese Information für die Suche überhaupt gegeben ist.

vordefinierte Sortierkriterien: Um eventuelle sinnlose Kombinationen von Sortierkriterien auszuschließen werden bestimmte Kombinationen von Suchkriterien vorgegeben, aus denen der Benutzer auswählen kann.

Die Unterdrückung der Synonymsuche hat sich aus dem Szenario ergeben, dass für einen bzw. mehrere Suchbegriffe durch die Einbeziehung von Synonymen die Übersichtlichkeit verloren geht, sowie dem Fall, dass es durch die Suche in der Synonymdatenbank zu Schleifen kommen kann.

In der Standardsuche werden aus dieser erweiterten Suchmaske die entsprechenden Optionen mit Defaultwerten belegt.

B. Benutzersicht

Der Benutzer soll über die Standardsuche oder die erweiterte Suche zur ersten Darstellung seiner Suchergebnisse geführt werden.
Um folgende Erläuterungen zu verstehen, werden hier zunächst ein paar Begriffe erklärt:

- Ergebnisgruppe: fasst mehrere Unterergebnisgruppen zusammen (z.B. eine Series umfasst mehrere Lecturegroups, zu einem Jahr gibt es mehrere Series, die in diesem Jahr gehalten wurden)
- Unterergebnisgruppe: fasst mehrere Lectures zusammen (z.B. eine Lecturegroup umfasst mehrere Lectures, aber auch ein Jahr umfasst mehrere Lectures)
- Ergebnis: ist eine Lecture
- Direkttreffer: Treffer eines direkt eingegebenen Suchwortes, kann ein Cluster- oder Einzelworttreffer sein
- Synonymtreffer: Treffer eines Synonyms eines eingegebenen Suchwortes, kann ein Cluster- oder Einzelworttreffer sein
- Clustertreffer: hat eine Länge in Sekunden- bis Minutenbereich, kann ein Direkt- oder Synonymtreffer sein und
- Einzelworttreffer: hat eine zeitlich ausgedehnte Länge im Millisekunden- bis Sekundenbereich, kann ein Direkt- oder Synonymtreffer sein

Die Unterscheidung der Treffer erfolgt auch grafisch:

- Direkttreffer, die Einzelworttreffer sind: Farbe1
- Direkttreffer, die Clustertreffer sind: Abstufung der Farbe1
- Synonymtreffer, die Einzelworttreffer sind: Farbe2
- Synonymtreffer, die Clustertreffer sind: Abstufung der Farbe2

Treffer haben zeitlich ausgedehnte Längen und damit können Ergebnisgruppen, Unterergebnisgruppen und Ergebnissen errechnete Längen zugeordnet werden:

- Gesamtlänge eines Ergebnisses: ergibt sich aus der Addition der Längen aller Treffer einer Lecture
- Gesamtlänge einer Unterergebnisgruppe: ergibt sich aus der Addition aller Gesamtlängen der in einer Unterergebnisgruppe enthaltenen Lectures
- Gesamtlänge einer Ergebnisgruppe: ergibt sich aus der Addition aller Gesamtlängen der in einer Ergebnisgruppe enthaltenen Unterergebnisgruppen

☑	Series5	47min
☑	Series1	43min
☐	Series19	6min

Fig. 7. Skizze von Seite 1 der Ergebnisdarstellung

In der ersten Darstellung wird das erste Sortierkriterium berücksichtigt, im Standardfall soll hier nach der Gesamtlänge der Ergebnisgruppen sortiert werden. Die Ansicht zeigt daher eine Art Tabelle (siehe Bild 7), in der Ergebnisgruppen, im Standardfall Titel von Vorlesungsreihen, mit der jeweils errechneten Gesamtlänge der Ergebnisgruppe dargestellt werden. Der Benutzer kann in dieser Ansicht, in der im Standardfall auch die Beschreibungen der Vorlesungsreihen angezeigt werden können, eine Vorauswahl darüber treffen, welche der Ergebnisgruppen detaillierter angesehen werden sollen und welche nicht. Über einen Button „Weiter“ wird der Benutzer weitergeleitet zur zweiten Seite der Ergebnisdarstellung, in der

nur noch die bereits getätigte Auswahl (mehrere Häkchen sind möglich) dargestellt wird; jede Auswahl (je „Häkchen“) soll dabei grafisch abgegrenzt werden von den übrigen, beispielsweise durch Bilden von Reitern wie in Abbildung 8.

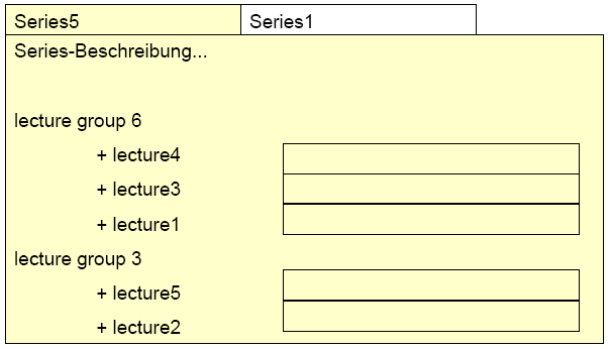

Fig. 8. Skizze von Seite 2 der Ergebnisdarstellung

Auf der zweiten Seite geht das zweite Sortierkriterium in die Darstellung ein, im Standardfall sind das die Lecturegroups. Je auf Seite Eins der Darstellung ausgewählter Ergebnisgruppe sollen nun die Unterergebnisgruppen mit den dazugehörigen Lectures, die die Treffer enthalten, angezeigt werden. Das bedeutet zu jeder Lecture gibt es einen Balken, in dem Treffer farblich dargestellt werden. Bei diesen Balken handelt es sich zunächst um eine Grobansicht (Bild ?? skizziert diese). Welche Treffer mit welcher dargestellt werden, wurde bereits weiter oben in diesem Kapitel genannt.

In dieser Grobansicht kann es dazu kommen, dass sich zwei Clustertreffer, die Treffer zwei verschiedener Synonyme sind, überschneiden, dies aber nicht sichtbar wird, da sie mit der gleichen Abstufung der Farbe für Synonyme dargestellt werden. Es kann auch dazu kommen, dass sich ein Clustertreffer eines Suchwortes mit einem Clustertreffer eines Synonyms überschneidet und dadurch Anfang und Ende eines der Cluster nicht sichtbar und damit zunächst aus Benutzersicht nicht korrekt dargestellt wird, aber der Benutzer soll hier auch nur einen Überblick über mögliche Treffer erlangen. Feinheiten werden in der zu jeder Lecture nachladbaren Detailansicht deutlich, wo zu jedem Suchwort und bei Synonymsuche auch zu jedem Synonym ein Balken mit der Unterscheidung zwischen Einzelwort- und Clustertreffern erstellt wird. Die Detailansichten soll der Benutzer per Klick baumartig aufklappen können (je Lecture) wie es viele Benutzer vom Microsoft Windows-Explorer kennen.

Die zweite Darstellungsseite soll zusätzlich noch einmal die Möglichkeit bieten, bestimmte Ergebnisse aus- bzw. einzublenden oder sogar zu verwerfen. Das betrifft die Unterscheidung zwischen Direktreffern und Synonymtreffern bzw. Einzelworttreffern und Clustertreffern in der Grobansicht, eventuell auch Medientypen. In der Detailansicht betrifft dies einzelne Detailbalken.

Letztendlich kann der Benutzer sich durch Klick auf Treffer in Detailbalken die entsprechenden Medien anzeigen lassen.

VI. Schnittstellen

Die eigentliche Suche stellt eine eigenständige Funktionalität dar und bildet daher eine Komponente innerhalb des Gesamtsystems. Um diese Komponente in eine Web-Seite einbinden zu können, muss dem entsprechend ein Interface angeboten werden. Da sich die Eingabeformen für die Standardsuche und die erweiterte Suche unterscheiden - Standardsuche über Eingabefeld und erweiterte Suche über eine komplette Seite -, werden sich die Schnittstellen voneinander unterscheiden. Während die Standardsuche als `ContentComponent` innerhalb einer Seite integriert wird, werden die erweiterte Suche und die Ergebnisseiten mit Hilfe der `PageController`-Klasse implementiert. Weitere Informationen finden sich in den projektbezogenen Ausarbeitungen ”Navigation” und ”CMS für dynamische Inhalte”.

Glossar

Cluster
: Definiertes Zeitintervall, in dem ein Wort gehäuft auftritt. Die Definition der Intervalllänge erfolgt durch die Produzenten des Datenbankinhalts.

Lecture
: Eine Vorlesungsveranstalung im Rahmen einer Lecture-Group.

Lecture-Group
: Eine Reihe von Vorlesungsveranstaltungen zu einem bestimmten Thema. Z.B. Einführung in die Technische Informatik.

Session
: Zeitintervall, während dem ein Benutzer sich auf einer Webseite bzw. einer Webplattform aufhält

REFERENCES

[1] SVG 1.1 specification. [Online]. Available: http://www.w3.org/TR/SVG/

[2] S. et al., “Ajax - konzept und anwendung.”

[3] The Smarty website. [Online]. Available: http://smarty.php.net/

[4] T. Joachims, “A statistical learning model of text classification for support vector machines,” online as technical paper.

[5] Webseite von Timon Schroeter mit Vortrag über SVMs vom 22C3. [Online]. Available: http://timon.schroeter.it/wiki/Wiki.jsp?page=22c3.pub

[6] The AdoDB website. [Online]. Available: http://adodb.sourceforge.net/

[7] The PHP website. [Online]. Available: http://www.php.net/

[8] GD Bibliothek auf www.php.net. [Online]. Available: http://www.php.net/gd

[9] Buchstabenerkennung (SVM Wettbewerbe). [Online]. Available: http://lmb.informatik.uni-freiburg.de/lectures/svm_seminar/wettbewerb.de.html

[10] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[11] M. S. Ageev and B. V. Dobrov, “Support vector machine parameter optimization for text categorization problems,” online as technical paper.

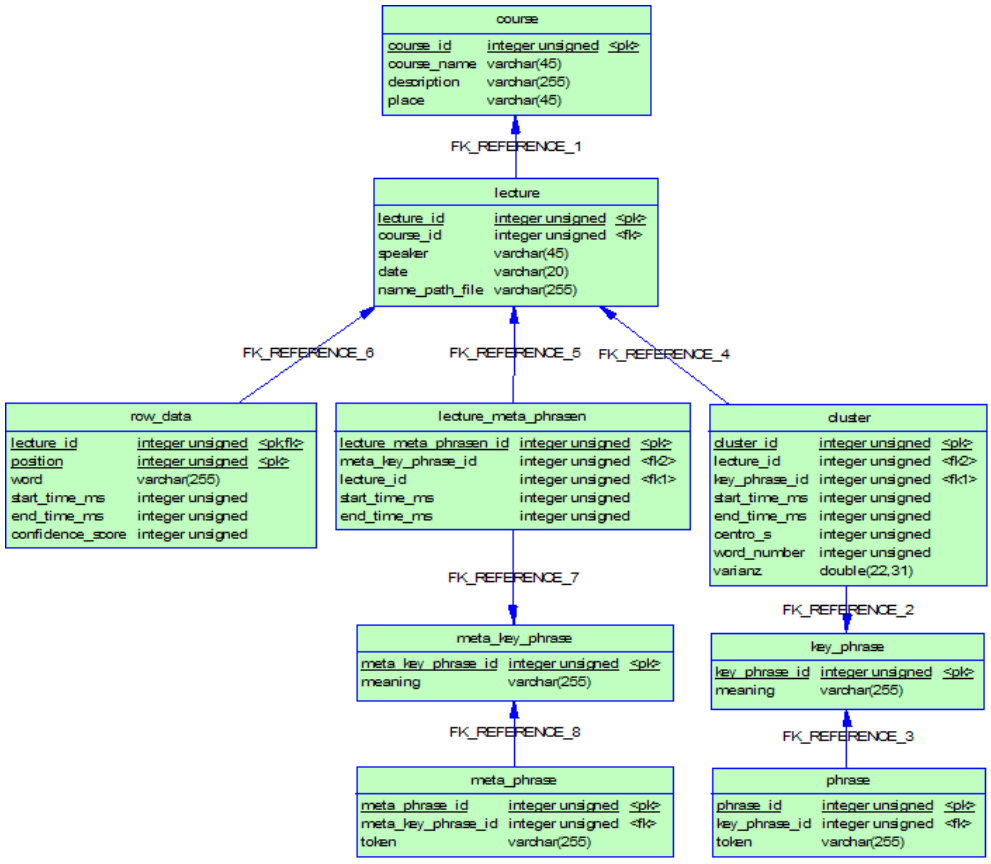

Fig. 2. Datenbankschema