

WhoKnows?

Seminararbeit im Seminar
SEMANTIC MULTIMEDIA
Sommersemester 2010
Hasso-Plattner-Institut Potsdam

vorgelegt von

Eyk Kny, Sebastian Kölle, Gerald Töpfer, Emilia Wittmers

13. Oktober 2010

Kurzzusammenfassung

Yovisto ist eine Internetplattform, die Nutzern akademische Video- und Vorlesungsaufzeichnungen zur Verfügung stellt. Dazu wird unter anderem eine auf semantischen Technologien basierende explorative Suche angeboten, mit der Anwender sich Schritt für Schritt ihrem Suchziel nähern können. Diese Suche nutzt dabei die Daten der freien DBpedia, um Beziehungen zwischen verschiedenen Suchbegriffen herzustellen. Allerdings benötigt die Suche komplexe Algorithmen und Heuristiken, die manuell evaluiert werden müssen. Um diesen Schritt einfacher und angenehmer zu gestalten wurde das browser-basierte Spiel *WhoKnows?* entwickelt. Während der Nutzer das Spiel spielt, werden Daten gesammelt, die zur Evaluation der Heuristiken verwendet werden können.

In dieser Arbeit wird die Spielidee, die Softwarearchitektur, sowie der Prozess von der Aufbereitung der DBpedia-Daten bis zur Auswertung der gesammelten Informationen beschrieben. Ein weiterer Schwerpunkt liegt auf der Integration des Spiels in das soziale Netzwerk Facebook, über das viele potentielle Nutzer erreicht werden können.

Inhaltsverzeichnis

1	Problemstellung: Explorative Suche in Yovisto	4
2	Lösungsansatz	5
2.1	Spielbasierte Erzeugung semantischer Daten	5
2.2	Die Weisheit der Vielen	5
2.3	WhoKnows? - Das Spielkonzept	6
3	Graphische Umsetzung des Spiels	11
3.1	Präattentive Wahrnehmung	11
3.2	Visualisierung der Spielvarianten	11
3.3	Visualisierung der Lösungen	12
3.4	Der 'Waitscreen'	13
3.5	Animationen	13
4	Architektur des Spiels	14
4.1	Überblick über die Architektur	14
4.2	Die Komponenten der Serverarchitektur	14
5	Aufbereitung der DBpedia-Daten	17
5.1	Verarbeitung großer Datenmengen mit Hadoop	17
5.2	Irrelevante Tripel herausfiltern	19
5.3	Distinkte Menge von Ressourcen ermitteln	19
5.4	Entitäten (Ober-)Kategorien zuordnen	20
5.5	Entitäten gewichten	21
6	Generierung von Fragen	22
6.1	Das Gewicht von Tripel und Prädikaten berechnen	22
6.2	Tripel den Schwierigkeitsgrad zuweisen	23
6.2.1	Eins-zu-Eins-Strategie	23
6.2.2	Eins-zu-N-Strategie	23
6.3	Generierung von falschen Antwortmöglichkeiten	24
6.3.1	Beste Kategorie des Subjekts ermitteln	24
6.3.2	Falsche Antwortmöglichkeiten auswählen	25
7	Auswertung von Spielen	26
7.1	Das Punktemodell	26
7.2	Erstellen von Spielstatistiken	27
7.3	Bewertung des Spiels durch den Benutzer	28
7.3.1	Anzeigen von Informationen zu Entities	28
7.3.2	Die Dislike-Funktion	30
8	Soziale Netzwerke	31
8.1	Facebook als soziales Netzwerk	31
8.2	Einbettung des Spiels in Facebook	32

9 Projekt Evaluation	34
9.1 Probleme und Anmerkungen	34
9.2 Erste statistische Daten aus der Nutzung	36
9.3 Konklusion	37
10 Future Work	38
Literaturverzeichnis	40

1 Problemstellung: Explorative Suche in Yovisto

Das Videoportal Yovisto ist eine Webanwendung aus dem Bereich 'E-Learning and Education'. Yovisto verwaltet akademische Videos und Vorlesungsaufzeichnungen, welche von den Nutzern des Portals nicht nur hochgeladen, angesehen, diskutiert und verschlagwortet, sondern auch inhaltlich durchsucht werden können.

Die inhaltsbasierte Suche über die semantischen Daten¹, die zu den Aufzeichnungen vorliegen, bietet dem Nutzer die Möglichkeit, sich in Yovistos großem und immer weiter wachsendem Videobestand zurechtzufinden und die richtigen Informationen zu dem von ihm gesuchten Thema zu erhalten.

Da es nicht selten vorkommt, dass ein Nutzer zu Beginn seiner Suche noch kein spezifisches Zieldokument im Sinn hat und oftmals noch nicht einmal genau weiß, nach welchen Begriffen er überhaupt suchen muss, um die gewünschten Informationen zu finden, gestaltet sich die Suche nach den Videoaufzeichnungen in Yovisto als eine explorative Suche [Mar06]: Der Nutzer erhält zu dem von ihm gesuchten Stichwort zum einen eine Liste der Videos, die dieses beinhalten, zum anderen aber auch eine Auflistung verwandter Suchbegriffe. Die von Yovisto vorgeschlagenen Begriffe sollen dem Nutzer dabei helfen, sein Suchziel schneller zu erreichen. Ist nämlich das gewünschte Video unter dem vom Nutzer angegebenen Schlagwort nicht zu finden, kann er einen der verwandten Begriffe verwenden, um seine Suche fortzusetzen.

Zur Erstellung der Liste verwandter Suchbegriffe nutzt Yovisto 'Linked Open Data'-Datenbestände wie die der DBpedia². Die DBpedia erfasst die strukturierten Informationen der weltweit größten Online-Enzyklopädie Wikipedia³ und stellt diese anderen Web-Anwendungen zur Verfügung.

Da in der DBpedia zu den gesuchten Begriffen häufig viel zu viele Daten zu finden sind, die für Yovistos explorative Suche zum Teil nicht einmal relevant sind, müssen die dort gelisteten Informationen zuerst gewichtet werden, bevor die wesentlichsten von ihnen dem Yovisto-Nutzer in einer überschaubaren Schlagwortliste präsentiert werden können. Die Gewichtung der Daten geschieht hierbei durch Heuristiken, die Waitelonis et al. in [WSHK10] detailliert beschreiben. Wie gut jedoch die Qualität der Ergebnisse dieser Heuristiken ist, konnte bisher noch nicht ausreichend evaluiert werden. Zwar stellen die Autoren ein Evaluierungsverfahren vor, mit dessen Hilfe ermittelt werden kann, inwiefern die explorative Suche den Nutzer dabei unterstützt, sein Suchziel zu erreichen, doch bleiben dabei die Fragen offen, ob durch die Heuristiken eventuell wichtige Daten unterschlagen oder irrelevante Daten als wichtig erachtet werden.

Wir haben uns mit der beschriebenen Problematik auseinandergesetzt und einen Lösungsansatz entwickelt, mit dem die Qualität der Heuristiken für die explorative Suche in Yovisto zukünftig besser evaluiert werden kann. Dieser Ansatz basiert auf dem Konzept der Spiele, die einen Nutzen für das Semantische Web besitzen, wie sie in dem Paper [SH08] von Katharina Siorpaes und Martin Hepp vorgestellt werden.

¹Semantische Daten: Informationen mit Bedeutung; Metadaten - über sie lassen sich Zusammenhänge schließen und Beziehungen knüpfen

²<http://dbpedia.org/About>

³<http://www.wikipedia.org/>

2 Lösungsansatz

2.1 Spielbasierte Erzeugung semantischer Daten

Der Mensch ist geübt darin, Dingen Bedeutungen zuzuweisen und Zusammenhänge zwischen diesen Dingen zu erkennen. Einem Menschen fällt es leicht, einer Abbildung Begriffe zuzuordnen, die mit dieser in Beziehung stehen, für einen Computer ist das jedoch eine sehr komplexe Aufgabe. Das automatische Erzeugen semantischer Daten ist schwierig und umfangreich. Ein großes Potenzial bietet in diesem Zusammenhang daher die Nutzung des Web 2.0: die Generierung semantischer Daten durch den Webanwender. Seit einigen Jahren gibt es Ansätze, nutzergenerierte Inhalte über Spiele erzeugen zu lassen. Luis von Ahn ist hier ein wichtiger Vorreiter. Er entwickelte gemeinsam mit Kollegen Spiele wie das 'ESP Game' [AD04] und 'Phetch' [AGK⁺06] zum Beschriften von Bildern, 'Peekaboom' [ALB06] zur Lokalisierung von Objekten in Abbildungen und 'Verbosity' [AKB06], ein Spiel mit dem über das Spielprinzip des Gesellschaftsspiels 'Tabu' allgemeingültige Fakten gesammelt werden können. All diese Ideen basieren auf demselben Konzept: Menschen spielen die Spiele als Unterhaltung und zum Zeitvertreib und erzeugen dabei als Nebeneffekt semantische Daten.

Die Ansätze dieses Konzepts nutzen auch wir in unserem Lösungsvorschlag zur Evaluierung der Heuristiken für die explorative Suche in Yovisto. Der Unterschied zwischen unserer Spielidee, die im Folgenden beschrieben werden soll, und den oben genannten Spielen liegt jedoch darin, dass es bei uns nicht darum geht, neue semantische Daten zu generieren, sondern bereits vorhandene Informationen zu überprüfen und zu evaluieren.

2.2 Die Weisheit der Vielen

Ob eine Information zu einem Objekt relevant oder unwichtig ist, kann jeder Mensch für sich persönlich subjektiv entscheiden. Ob diese Information deshalb aber auch objektiv betrachtet eine relevante Eigenschaft für dieses Objekt ist, ist durch die Aussage eines Einzelnen noch nicht geklärt. Um eine objektive Einschätzung zu erhalten, muss die Meinung vieler betrachtet werden, die unabhängig voneinander dieselbe Information beurteilen und je nachdem zu ähnlichen oder abweichenden Ergebnissen gelangen. Unser Lösungsansatz zur Evaluierung der Heuristiken für die explorative Suche in Yovisto basiert auf der Annahme, dass wenn eine sehr große Menge von Menschen derselben Meinung ist, diese auch als allgemeingültig anerkannt werden kann.

In seinem Buch 'The Wisdom of Crowds' [Sur04] erklärt James Surowiecki, wie unter der Einhaltung gewisser Kriterien durch Gruppenentscheidungen bessere Problemlösungen für Aufgabenstellungen gefunden werden können als durch das Wissen von Einzelnen. Die Kriterien, die er dabei nennt, sind die folgenden vier Punkte:

1. Meinungsvielfalt
2. Unabhängigkeit
3. Dezentralisierung
4. Aggregation

Diese Kriterien werden in unserem Problemfall in jedem Punkt erfüllt, da das Spiel, welches wir als unseren Lösungsansatz im folgenden Abschnitt vorstellen werden, als ein Spiel konzipiert ist, das in sozialen Netzwerken wie Facebook (das zurzeit größte soziale Netzwerk weltweit) gespielt wird.

Die Gruppenmitglieder solcher Netzwerke unterscheiden sich in vielen ihrer Eigenschaften wie Geschlecht, Alter und Herkunft und besitzen verschiedene Informationen über die Begriffe, die bei der explorativen Suche in Yovisto auftauchen. Diese differierenden Voraussetzungen führen zu einer großen Meinungsvielfalt bei den Nutzern über die Wichtigkeit von Objekteigenschaften der zu betrachtenden Begriffe.

Die Tatsache, dass unser Spiel so aufgebaut ist, dass die Spieler dazu angehalten werden, ihr Wissen für sich zu behalten, um einen möglichst hohen individuellen Punktestand zu erreichen, gewährleistet die Unabhängigkeit der Meinungen.

Wenn ein Nutzer Gefallen an unserem Spiel gefunden hat, ist er motiviert, einen höheren Punktestand als zuvor zu erreichen und wird das Spiel ein weiteres Mal spielen wollen. Je öfter er es spielen wird, desto spezialisierter wird sein Wissen sein und desto höher ist sein Ansporn, sich von außerhalb weiteres Spezialwissen anzueignen (siehe hierzu Kapitel 7.3.1). Das Interesse, gestellte Probleme lösen zu wollen, fällt unter Surowieckis Kriterium der Dezentralisierung.

Die Daten, die durch das Spielen der Nutzer erzeugt werden, werden in einer Datenbank gespeichert. Über die so gesammelten Informationen lassen sich die Aussagen der verschiedenen Spieler zu einer Gesamtmeinung aggregieren.

Da alle vier Kriterien nach Surowiecki erfüllt sind, kann festgehalten werden, dass die Nutzung der 'Weisheit der Vielen' in unserem Fall eine geeignete Herangehensweise ist, die nicht zu irrationalen und fehlerhaften Entscheidungen führt.

2.3 WhoKnows? - Das Spielkonzept

Unser Lösungsansatz zur Evaluierung der Heuristiken für die explorative Suche in Yovisto ist das Spiel *WhoKnows?*.

WhoKnows? ist ein *Semantic Web Game*, das im Einzelspieler-Modus in sozialen Netzwerken gespielt wird. Das Ziel für den Nutzer ist es, in dem Spiel so viele Punkte wie möglich zu erreichen, um besser abzuschneiden als alle anderen Netzwerk-Teilnehmer.

In *WhoKnows?* werden dem Spieler Fragen zu Eigenschaften von Begriffen gestellt, welche er best- und schnellstmöglich beantworten muss. Fragen zu Objekteigenschaften, die von sehr vielen Spielern sehr oft richtig beantwortet werden, deuten darauf hin, dass diese Eigenschaften bei der Allgemeinheit der Menschen bekannt und somit wichtig für das Objekt sind. Diese Eigenschaften sollten solche sein, die auch von den Heuristiken als wichtig erkannt werden. Korreliert dies nicht miteinander, ist es ein Hinweis darauf, dass die Heuristiken eventuell wichtige Begriffe für die explorative Suche unterschlagen. Umgekehrt gehen wir davon aus, wenn Fragen zu Objekteigenschaften von sehr vielen Spielern sehr oft unterschiedlich beantwortet werden, dass diese dann auch allgemein eher unbekannt sind und dadurch vermutlich keine relevanten Eigenschaften sein können. Die Fragen zu den Objekteigenschaften in *WhoKnows?* werden aus den Daten der DBpe-

dia erstellt. Eine Frage mit Antwort besteht dabei immer aus dem Lösungstripel <Subjekt, Prädikat, Objekt> und kann zusätzlich je nach Spielvariante und Schwierigkeitsgrad weitere Subjekte als Antwortmöglichkeiten vorgeben. Ein Beispiel hierfür ist die Frage nach der Objekteigenschaft 'Sprache' für das Subjekt 'Chile'. Diese würde aus dem DBpedia Tripel <chile, language, spanish language> gebildet werden (siehe Abbildung 1) und dementsprechend wie folgt lauten: 'Spanish language is a/ the language of' mit der Antwortmöglichkeit 'Chile' (siehe Abbildung 2).

Im Gegensatz zu Spielen, die dazu genutzt werden neue semantische Inhalte zu generieren, lässt sich bei *WhoKnows?* die Richtigkeit einer gegebenen Spieler-Antwort leicht überprüfen. Die Antworten auf die Fragen liegen in der DBpedia bereits vor und müssen nicht, wie beispielsweise in den von Luis von Ahn vorgestellten Spielen, von einem unabhängigen Spieler gegengeprüft werden.

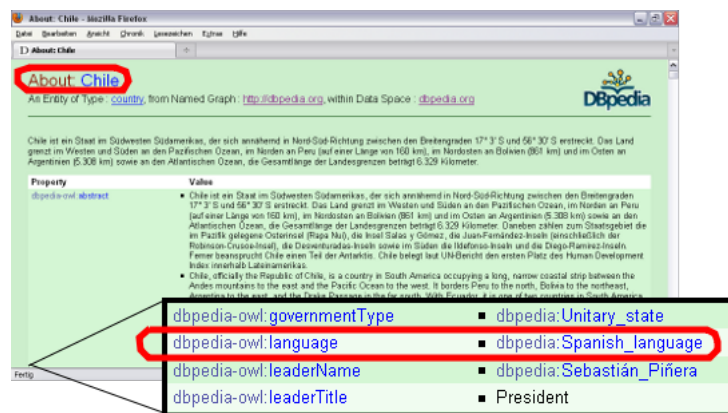


Abbildung 1: Screenshot der DBpedia zu dem Stichwort 'Chile'



Abbildung 2: Screenshot einer Eins-zu-Eins-Frage

Der Spielverlauf Ruft ein Spieler *WhoKnows?* (zum Beispiel über seinen Facebook-Account) auf, öffnet sich das Hauptmenü. Hier hat er die Möglichkeiten, sofort mit dem Spielen zu beginnen oder sich die Highscore-Liste aller Spieler des sozialen Netzwerkes anzusehen, die es unter die besten Zehn geschafft haben.

Der Spieler hat die Optionen, ein neues Spiel zu starten oder das letzte von ihm nicht zu Ende gespielte Spiel fortzusetzen.

Wählt der Spieler ein neues Spiel, beginnt er im ersten Level mit null Punkten und fünf Leben. Das Level steht hier für den Schwierigkeitsgrad und wird erhöht, wenn der Spieler die Fragen richtig beantwortet und verringert, wenn die Antworten nicht korrekt sind (solange der Spieler nicht bereits im ersten Level ist).

Die Leben des Spielers zeigen an, wie viele falsche Antworten er geben kann bis das Spiel beendet ist. Für falsche Antworten werden ihm Leben abgezogen. Hat er keine Leben mehr zur Verfügung, ist das Spiel vorbei.

Dem Spieler werden nacheinander verschiedene Fragen gestellt, die er in einer vorgegebenen Zeit zu beantworten hat. Nach jeder Frage erhält er eine Rückmeldung, ob seine Antwort richtig oder falsch gewesen ist. Je nachdem, wie korrekt seine Lösung war und wie schnell er sie gegeben hat, werden seine Punkte erhöht oder verringert.

Wie die Fragen, die dem Spieler gestellt werden, im Detail aussehen, hängt von der Spielvariante ab, welche im folgenden Abschnitt des Dokuments erläutert werden. Das Erscheinen der jeweiligen Spielvariante ist aufgrund der verschiedenen Schwierigkeitsgrade abhängig von dem aktuellen Level.

Eins-zu-Eins-Variante, einfach Die einfache Eins-zu-Eins-Variante (siehe Abbildung 2) ist der simpelste Spieltyp in *WhoKnows?*. Nach dem Konzept von Multiple Choice Aufgaben wird dem Spieler zu der gestellten Frage eine Liste von Antwortmöglichkeiten vorgegeben, aus der er eine einzige Lösung auszuwählen hat. Die Anzahl der vorgegebenen Antworten ist dabei abhängig von dem Level, in dem sich der Spieler gerade befindet. Von Level eins bis drei erhält er zwei Lösungsvorschläge, von Level vier bis sechs stehen drei Antworten zur Auswahl, von Level sieben bis neun bekommt er vier Antwortmöglichkeiten vorgeschlagen usw. Die Erhöhung in Dreier-Schritten wird so fortgeführt, bis ab einer Liste von acht Vorschlägen die Anzahl nicht mehr inkrementiert wird.

Eins-zu-Eins-Variante, Hangman Bei der Hangman-Variante (siehe Abbildung 3a) des Eins-zu-Eins-Typs werden keine Antwortmöglichkeiten vorgegeben. Der Spieler muss hier die Lösung nach dem Spielprinzip von Hangman erraten. Er darf dabei bis zu fünfmal einen falschen Buchstaben angeben; beim sechsten Fehlversuch wird seine Antwort als ungültig gewertet. Die maximale Länge des zu erratenden Wortes ist levelabhängig und berechnet sich nach der folgenden Formel: $\# \text{Buchstaben} = \# \text{Level} + 4$.

Die Hangman-Variante kann ab dem dritten Level auftauchen. Vor dem dritten Level werden nur Fragen der einfachen Eins-zu-Eins-Variante gestellt.

Eins-zu-Eins-Variante, Formula Eine andere Variante des Eins-zu-Eins-Spieltyps ist 'Formula' (siehe Abbildung 3b), bei der, wie bei der einfachen Variante, dem Spieler eine gewisse Anzahl von Antwortmöglichkeiten vorgegeben wird.

Der Unterschied zu dem simplen Spieltyp besteht hier darin, dass neben den Antworten arithmetische Aufgaben stehen, die zu berechnen sind. Der Spieler muss die Lösung der Aufgabe errechnen, die neben der Antwort steht, die er für die richtige hält, und diese in das dafür vorgesehene Feld über die Tastatur eingeben.

Die Anzahl der aufgelisteten Antwortmöglichkeiten ergibt sich bei der mathematischen Spielvariante nach dem gleichen Prinzip wie bei dem einfachen Eins-zu-Eins-Spiel, nur dass hier maximal sechs Antworten zur Auswahl stehen können. 'Formula' kann erst ab dem fünften Level gespielt werden und erscheint von da an zufällig.

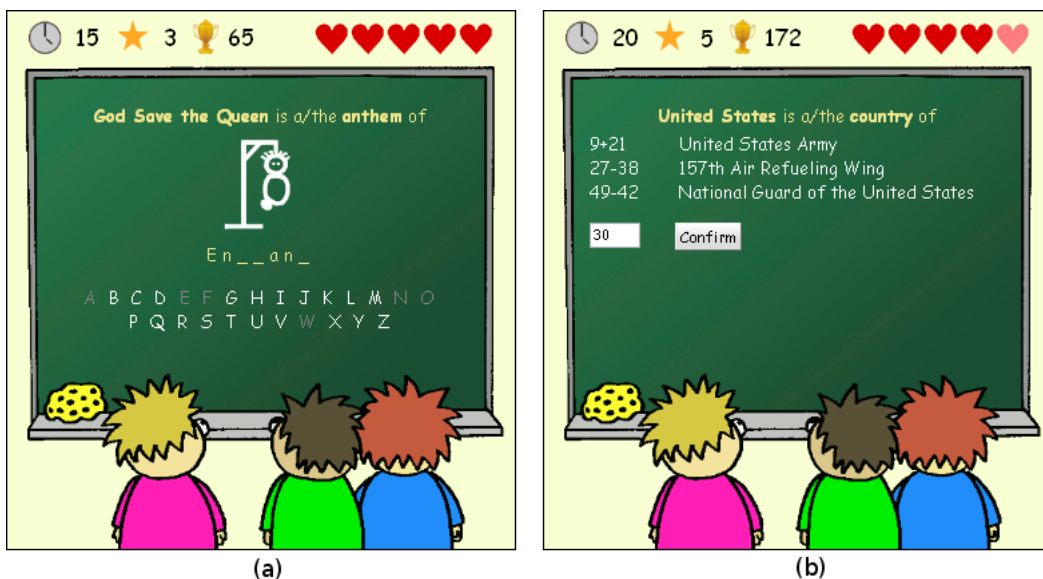


Abbildung 3: a) Screenshot, Hangman-Variante b) Screenshot, mathematische Variante

Eins-zu-N-Variante, einfach In der Eins-zu-N-Variante von *WhoKnows?* (siehe Abbildung 4) erhält der Spieler wie in der Eins-zu-Eins-Variante eine Liste von Antwortmöglichkeiten. Bei dem Eins-zu-N-Spiel kann es jedoch mehr als nur eine richtige Lösung geben, sodass der Spieler auch mehrere Antworten auswählen darf.

Diese Variante wird ab dem siebten Level freigegeben. Von Level sieben bis neun stehen zwei Lösungsvorschläge zur Auswahl, von Level zehn bis zwölf drei usw. Die Anzahl der Antworten wird solange in Dreier-Schritten erhöht bis die Grenze von acht erreicht ist.



Abbildung 4: Screenshot der Eins-zu-N-Variante

3 Graphische Umsetzung des Spiels

Bevor wir im Folgenden die graphische Umsetzung des Spiels erläutern wollen, soll zunächst einmal der Begriff 'präattentive Wahrnehmung' vorgestellt werden, da die Erkenntnis, die aus dieser Form der Wahrnehmung folgt, nicht ganz unerheblich für die Visualisierung ist.

3.1 Präattentive Wahrnehmung

Als präattentive Wahrnehmung wird eine vorbewusste Perzeption⁴ bezeichnet, die zwar als Reiz vom Nervensystem registriert wird und dort eine Reaktion verursacht, jedoch nicht ins menschliche Bewusstsein dringt. Die präattentive Wahrnehmung funktioniert sehr schnell (innerhalb von 200 bis 250 Millisekunden), aber auch sehr akkurat.

Visuell lassen sich die graphischen Variablen wie Form, Größe, Anzahl, Farbe und Helligkeitswert sehr gut präattentiv wahrnehmen. Bei der Verwendung dieser Variablen ist jedoch zu beachten, dass sie miteinander interferieren können und somit in Kombination eventuell kaum noch oder gar nicht mehr vorbewusst zu erfassen sind. Als Beispiel für dieses Phänomen soll die folgende Abbildung dienen, in der sichtbar wird, dass die Wahrnehmung der Farbe die der Form überlagert:

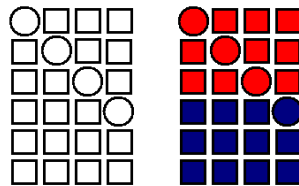


Abbildung 5: Interferenz visueller Variablen bei präattentiver Wahrnehmung

Die Verwendung präattentiv wahrnehmbarer Variablen ist dementsprechend nur dann sinnvoll, wenn man sie in einer geeigneten und angemessenen Weise einsetzt.

3.2 Visualisierung der Spielvarianten

Bei der Beantwortung der Fragen in *WhoKnows?* hat der Spieler nur eine recht kurze Zeitspanne zur Verfügung, in der er zu entscheiden hat, welche der Antworten er für die richtige(n) hält. Es ist daher besonders wichtig, dass der Spieler sofort weiß, um welche Spielvariante es sich bei der aktuellen Frage handelt.

Um die Eins-zu-N-Spielvariante (siehe Abbildung 4) von der Eins-zu-Eins-Variante (siehe Abbildung 2), welche vom Aufbau her beide identisch sind, graphisch voneinander abgrenzen zu können, haben wir zwei visuelle Variablen verwendet:

1. Hintergrundfarbe
2. Button-Form

⁴Perzeption:= die Gesamtheit aller Wahrnehmungsprozesse

Die Unterscheidung hinsichtlich der Hintergrundfarbe lässt sich sehr gut präattentiv wahrnehmen. Bei der Eins-zu-Eins-Variante wird die Frage auf einer grünen Tafel dargestellt, wohingegen die Frage einer Eins-zu-N-Variante auf einer blauen Tafel abgebildet wird.

Die Differenzierung bei der Form der Buttons unterstützt den Spieler bei seiner Entscheidungsfindung, ob nur eine einzige Lösung möglich ist oder auch mehrere Antworten richtig sein können. Hierbei wird das allgemein bekannte Konzept der 'Radio Buttons' und 'Check Boxen' verwendet. Sind die Buttons rund, gibt es nur eine Auswahlmöglichkeit; sind die Buttons eckig, sind mehrere Lösungen erlaubt. Menschen, die einen häufigen Umgang mit Computern haben, haben das Button-Konzept bereits stark verinnerlicht und nehmen so den Unterschied nicht mehr bewusst, sondern vorbewusst wahr.

3.3 Visualisierung der Lösungen

Hat der Spieler eine Frage beantwortet, erhält er sofort im Anschluss eine Rückmeldung, ob er diese korrekt gelöst hat.

Damit der Spieler schnell erfassen kann, ob seine Antwort richtig gewesen ist, wird das Ergebnis farblich gekennzeichnet. Hat der Spieler eine korrekte Antwort gegeben, wird diese grün hinterlegt. Ist seine Antwort falsch gewesen, wird sie rot markiert. Gelb gekennzeichnete Antworten sind die Lösungen, die richtig gewesen wären, die der Spieler jedoch nicht ausgewählt hat.

Zusätzlich zu der farbigen Kennzeichnung der Antworten nutzen wir auch das Hintergrundbild, um dem Spieler sein Ergebnis zu visualisieren. Da Menschen Gesichter sehr gut präattentiv wahrnehmen können, lässt sich nämlich auch über die Mimik der Spielmaskottchen eine Aussage über das Abschneiden des Spielers bei der zuletzt gestellten Frage treffen: Gucken die Figuren traurig, hat der Spieler etwas falsch gemacht; freuen sie sich jedoch, so war die Lösung richtig. (Siehe Abbildungen 6)



Abbildung 6: a) Screenshot, Falsche Antwort b) Screenshot, Richtige Antwort

3.4 Der 'Waitscreen'

Vor jeder Frage erscheint der 'Waitscreen', welcher dem Spieler drei Sekunden Zeit gibt, sich auf seine nächste Aufgabe vorzubereiten. Er bekommt an dieser Stelle die Möglichkeit, sich die Spielerklärung der folgenden Spielvariante anzusehen. Die Erklärung beinhaltet den Namen der Spielvariante, eine knappe textuelle Erläuterung (für verbale Lerntypen) und eine bildliche Darstellung (für visuelle Lerntypen). Während sich der Spieler die Spielanleitung ansieht, werden die drei Sekunden angehalten, damit er für das Verstehen der Aufgabe auch wirklich so viel Zeit zur Verfügung hat, wie er benötigt. (Siehe Abbildungen 7)

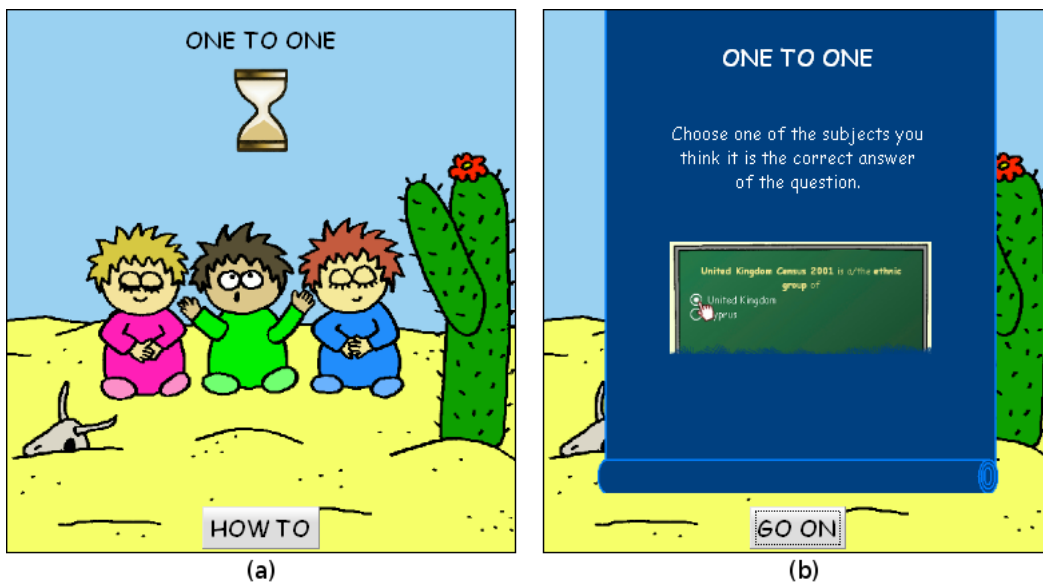


Abbildung 7: a) Screenshot, Waitscreen b) Screenshot, Waitscreen mit Spielerklärung

3.5 Animationen

Die Animationen in *WhoKnows?*, wie z.B. die Sanduhr-Animation im 'Waitscreen' (siehe Abbildung 7a), haben zwar keinen effektiven Nutzen für den Spielverlauf, jedoch lockern sie das Spiel mit seinen häufig doch recht trockenen Fragen ein wenig auf.

Da wir uns bei der Implementierung des Spiels für das Google Web Toolkit⁵ entschieden haben, nutzen wir für die Animation der Informationsanzeigen (siehe Abbildung 7b) GWT-FX⁶ und ansonsten animierte GIFs⁷.

Neue GIFs lassen sich dem Spiel über den von uns implementierten `RandomImageGenerator` sehr leicht hinzufügen.

⁵<http://code.google.com/intl/de-DE/webtoolkit/>

⁶<http://code.google.com/p/gwt-fx/>

⁷http://de.wikipedia.org/wiki/Graphics_Interchange_Format#Animationen

4 Architektur des Spiels

Im Folgenden wird ein Überblick über die für das Spiel entworfene Architektur gegeben. Als Anforderungen standen vor allem eine gute Wart- und Erweiterbarkeit (insbesondere im Hinblick auf neue Spieltypen) im Vordergrund.

4.1 Überblick über die Architektur

Die Grundidee der Architektur ist die Aufteilung der Anwendung auf einen clientseitigen und einen serverseitigen Teil⁸ (siehe Abbildung 8). Die Serveranwendung läuft dabei in einem Java Application Server, während die Clientanwendung als JavaScript im Browser des Nutzers ausgeführt wird. Dies ermöglicht es dem Nutzer, das Spiel auf jedem Rechner mit Internetzugang zu spielen, wobei gleichzeitig auf proprietäre Technologien wie Adobe Flash verzichtet werden kann.

Für die technische Umsetzung dieser Architektur wird das *Google Web Toolkit*⁹ (GWT) verwendet, das ein einheitliches Programmiermodell sowohl für die Client- als auch für die Serveranwendung bietet. Beide Anwendungen werden in Java entwickelt, wobei die Clientanwendung automatisch in JavaScript übersetzt wird. Die Kommunikation der Anwendungen wird dabei durch Remote-Procedure-Calls (RPC, [CD88, S. 197]) realisiert.

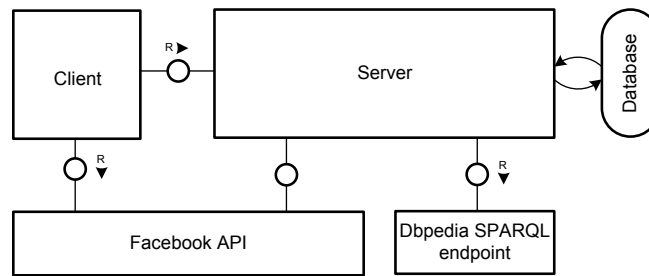


Abbildung 8: Überblick über die Architektur

Zusätzlich zur Integration von Client und Server, müssen auch externe Quellen, die Facebook API sowie der DBpedia SPARQL-Endpoint, eingebunden werden (vgl. Abschnitt 8.2 und 7.3.1). Für den Zugriff auf den DBpedia SPARQL-Endpoint wird die Bibliothek *Jena ARQ*¹⁰ verwendet.

4.2 Die Komponenten der Serverarchitektur

Abbildung 9 zeigt einen Überblick über die serverseitige Architektur. Um die Anforderungen gute Erweiterbarkeit und Wartbarkeit zu erfüllen, ist der Servercode in austauschbare Komponenten aufgeteilt. Für diesen Zweck wird das Dependency-Injection-Framework *Google Guice*¹¹ eingesetzt. Es ermöglicht, den Programmcode in verschiedene Komponenten zu unterteilen und deklarativ Abhängigkeiten zwischen diesen festzulegen. Mit

⁸ *Client-Server-Architektur*, vgl. [CD88, S. 35]

⁹ <http://code.google.com/webtoolkit/>

¹⁰ <http://jena.sourceforge.net/ARQ/>

¹¹ <http://code.google.com/p/google-guice/>

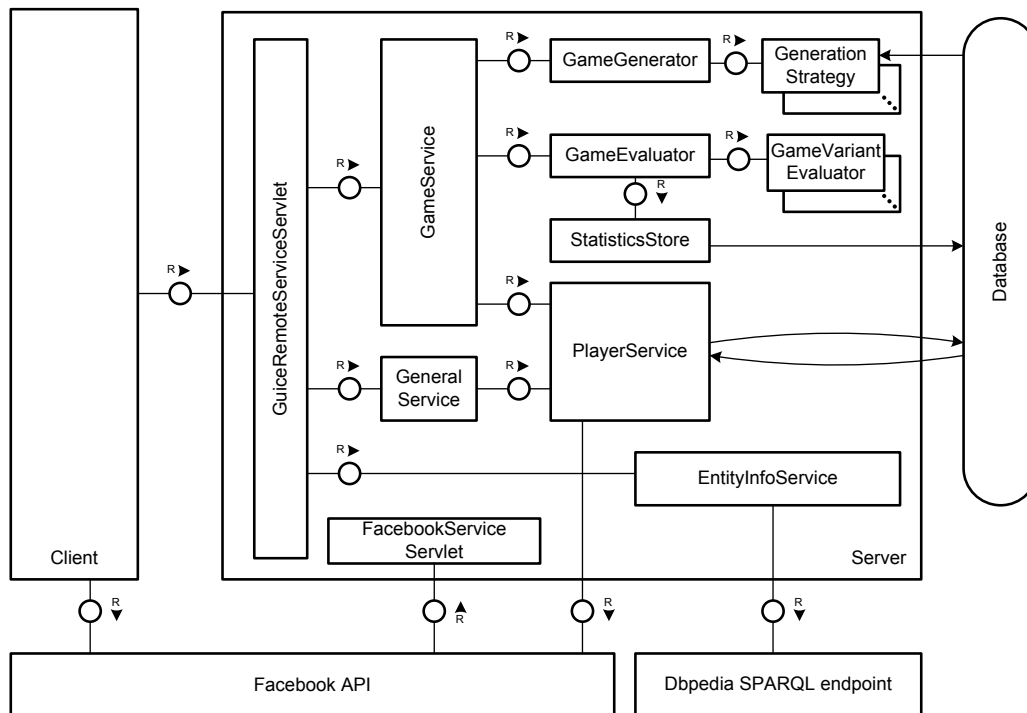


Abbildung 9: Übersicht über die Serverarchitektur

Hilfe von Modulen kann dann konfiguriert werden, welche konkreten Implementierungen der Komponenten in welchen Fällen verwendet werden sollen.

Im Folgenden werden die wichtigsten Serverkomponenten erläutert:

GuiceRemoteServiceServlet Das `GuiceRemoteServiceServlet` verbindet GWT mit Guice, indem es RPCs vom Client entgegennimmt und an den gewünschten GWT-Service weiterleitet.

GameService Der `GameService` bietet alle Methoden für den Client an, die zum Starten und Auswerten eines Spiels benötigt werden.

GeneralService Der `GeneralService` stellt allgemeine Funktionen für das Spiel bereit, beispielsweise die Abfrage der Highscoreliste.

EntityInfoService Der `EntityInfoService` erlaubt es dem Client, für jedes Entity einen kurzen beschreibenden Text anzufordern (vgl. Abschnitt 7.3.1).

FacebookServiceServlet Das `FacebookServiceServlet` nimmt Aufrufe von Facebook im Rahmen der Authentifizierung von Spielern entgegen (vgl. Abschnitt 8.2).

PlayerService Der `PlayerService` erlaubt es, den in der aktuellen Session aktiven Spieler abzufragen, sowie die geänderten Spielerdaten nach einem Spiel zu speichern.

GameGenerator Der **GameGenerator** generiert ein neues Spiel, indem eine **GenerationStrategy** ausgewählt und zusätzliche Parameter (zum Beispiel die Spielzeit) festlegt.

GenerationStrategy Eine **GenerationStrategy** legt fest, wie die Generierung für einen bestimmten Spieltyp (zum Beispiel Eins-zu-Eins-Spiele) abläuft. Dies beinhaltet beispielsweise die Auswahl von richtigen und falschen Antworten.

GameEvaluator Die Aufgabe des **GameEvaluator** besteht darin, vom Spieler gelöste Aufgaben mithilfe des richtigen **GameVariantEvaluators** zu bewerten.

GameVariantEvaluator Ein **GameVariantEvaluator** implementiert den Algorithmus zur Auswertung eines bestimmten Spieltyps. Dieser umfasst die Vergabe von Punkten und die Anpassung des aktuellen Levels sowie der Anzahl der Leben.

StatisticsStore Der **StatisticsStore** zeichnet die Ergebnisse für alle gelösten Fragen auf, um diese später für die Bestimmung der wichtigsten Properties zu verwenden (vgl. Abschnitt 7.2). Weiterhin werden vom Spieler mit *dislike* markierte Spiele zur späteren Überprüfung gespeichert (vgl. Abschnitt 7.3.2).

Die Erweiterbarkeit der Architektur ergibt sich insbesondere durch die **GameGenerators** und **GameEvaluators**. Denn um einen neuen Spieltyp zu integrieren, muss serverseitig nur jeweils eine neue Klasse implementiert werden.

5 Aufbereitung der DBpedia-Daten

Wie bereits erwähnt wurde, dient die DBpedia als Wissensbasis für die explorative Suche von Yovisto und somit auch für *WhoKnows?*. Die DBpedia ist ein Projekt zur Extraktion von strukturierten Informationen aus der Wikipedia, welches gemeinsam von der Universität Leipzig, der Freien Universität Berlin und dem Unternehmen OpenLink Software durchgeführt wird. Aktuelle Datensätze werden in regelmäßigen Abständen von der DBpedia zur Verfügung gestellt.¹² Grundlage zur Beschreibung der Daten bilden das 'Resource Description Framework'¹³ und die 'Web Ontology Language'¹⁴ (OWL), welche verbreitete Standards zur formalen Beschreibung von Informationen über Entitäten sind. Diese Informationen sind, wie im Abschnitt 2.3 beschrieben, in sogenannte Tripel (Turtle-Syntax¹⁵) abgelegt, bestehend aus Subjekt, Prädikat und Objekt. Durch die formale Repräsentation können die Informationen sehr leicht von Maschinen verarbeitet und ausgewertet werden.

Als Grundlage für die Entwicklung des Spiels wurde die DBpedia 3.5.1 verwendet. Diese Version umfasst insgesamt ca. eine Milliarde Tripel und stellt somit eine gigantische Menge von Informationen bereit. Viele dieser Tripel sind jedoch in diesem Kontext nicht von Bedeutung und können ignoriert werden. Da die Benutzerschnittstelle des Spieles auf Englisch gehalten werden sollte, um möglichst Vielen das Spielen von *WhoKnows?* zu ermöglichen, haben wir uns für die englische Version der DBpedia entschieden. Die für *WhoKnows?* relevanten (englischen) Datensätze sind:

- *Ontology Infobox Properties*, beinhaltet Eigenschaften von Entitäten.
- *Titles*, enthält textuelle Beschreibungen (Labels) zu den Ressourcen.
- *Articles Categories*, umfasst Ressourcen und deren direkte Kategorien.
- *Categories (Skos)*, enthält Oberkategorie- bzw. Unterkategorie-Beziehungen.
- *Pagelinks*, beinhaltet Verweise zwischen Wikipedia-Seiten.

Des Weiteren stellte das Yovisto-Team eine Liste mit Entitäten (im Folgenden als *Yovisto-Entitäten* bezeichnet) zur Verfügung, die für die explorative Suche von Bedeutung sind.

Auf Basis der genannten Datensätze wurden weitere Schritte durchgeführt, um die Menge der irrelevanten Daten weiter zu minimieren, implizite Informationen abzuleiten und in Folge dessen die Generierung von Fragen (siehe Kapitel 6) zu beschleunigen.

5.1 Verarbeitung großer Datenmengen mit Hadoop

Bevor die einzelnen Schritte (siehe Abbildung 11) genauer erläutert werden, soll an dieser Stelle ein Überblick über das Framework und die Infrastruktur, mit der die Datenaufbereitung durchgeführt wurde, gegeben werden.

¹²<http://wiki.dbpedia.org/Downloads>

¹³<http://www.w3.org/RDF>

¹⁴http://www.w3.org/2007/OWL/wiki/OWL_Working_Group

¹⁵<http://www.w3.org/TeamSubmission/turtle/>

Da die Vorverarbeitung der Daten (ca. 22 GB mit 151 Millionen Tripel) sehr viel Rechenzeit in Anspruch nimmt und somit auf einem einzelnen Desktop-Rechner kaum zu bewältigen ist, wurde für die Datenaufbereitung Hadoop¹⁶ verwendet. Beispielsweise kann die Zuordnung von Kategorien zu Entitäten nur mit sehr vielen I/O-Zugriffen in polynomieller Zeit gelöst werden. Auf einem einzelnen Desktop-Rechner müsste, aufgrund des zu kleinen Hauptspeichers, sehr oft auf die Festplatte zugegriffen werden. Da in einem Rechner-Cluster in der Summe weit aus mehr RAM zur Verfügung steht, kann die Verarbeitung von großen Datenmengen weitaus effizienter durchgeführt werden. Hadoop ist ein in Java geschriebenes Framework zur effizienten Verarbeitung großer Datenmengen auf Computerclustern. Viele bekannte Unternehmen, wie Amazon, Facebook, Google, IBM, u.a. nutzen Hadoop beispielsweise zur Auswertung von Log-Dateien, Erstellung von Suchindizes und Anwendung von verschiedenen Data-Mining-Verfahren.¹⁷ Grundlage des Frameworks bildet das von Google entworfene MapReduce-Paradigma[DG04] (siehe Abbildung 10).

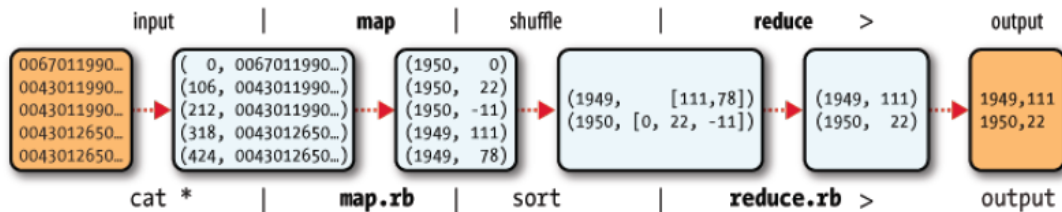


Abbildung 10: Der MapReduce-Algorithmus: Eingabe auf Schlüssel-Wert-Paare abbilden, Ausgaben nach Schlüssel sortieren und anschließend Werte aggregieren.

Für die Ausführung konnte ein Cluster mit zehn handelsüblichen Desktop-Rechnern vom Lehrstuhl Informationssysteme des Hasso-Plattner-Instituts (HPI) genutzt werden. Die komplette Datenaufbereitung benötigte ca. drei Stunden, wobei die Zuordnung von (Ober-)Kategorien zu den Entitäten (siehe Abschnitt 5.4) mit ca. zwei Stunden die meiste Zeit in Anspruch nahm. Als Alternative zum HPI-Cluster kann z.B. Amazons *Elastic MapReduce Service*¹⁸ genutzt werden. Bei einer ähnlichen Konfiguration sind hier jedoch 4 USD pro Stunde fällig.¹⁹

¹⁶<http://hadoop.apache.org/>

¹⁷<http://wiki.apache.org/hadoop/PoweredBy>

¹⁸<http://aws.amazon.com/de/elasticmapreduce/>

¹⁹<http://aws.amazon.com/de/elasticmapreduce/pricing/>

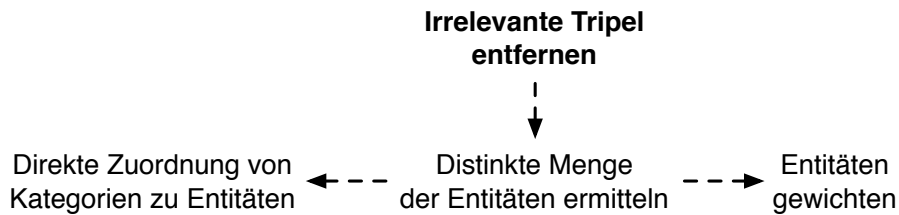


Abbildung 11: Schritte, die bei der Vorverarbeitung der Daten ausgeführt werden.

5.2 Irrelevante Tripel herausfiltern

Der Datensatz *Ontology Infobox Properties* enthält die elementaren Tripel, die als Grundlage für die Erstellung von Fragen dienen. Tripel, die beispielsweise die Nationalität von Personen, die Hauptstädte von Ländern oder die Sprache von Filmen beschreiben, sind in diesem Datensatz zu finden. Viele dieser Eigenschaften von Entitäten werden durch Literale beschrieben, wie z.B. Geburtsdaten, die Auflösung und Beschreibung eines Films oder die Breite eines digitalen Bildes. Oftmals sind diese Literale jedoch sehr lang oder zu speziell, so dass hier Probleme bei der Darstellung und dem Verständnis der Fragen zu erwarten sind. Wegen den genannten Problemen, aber auch weil bei der explorativen Suche von Yovisto ausschließlich Beziehungen zwischen Entitäten verwendet werden, haben wir uns dafür entschieden, nur Tripel zu betrachten, bei denen sowohl das Subjekt als auch das Objekt Yovisto-Entitäten sind (siehe Abbildung 12).

```

yovistoEntities ← GETYOVIStoENTITIES()

comment: key: line number; value: RDF triple
procedure MAP(key, value)
  subject ← GETSUBJECT(value)
  object ← GETOBJECT(value)

if subject ∈ yovistoEntities and object ∈ yovistoEntities
  then EMIT(value, null)
  
```

Abbildung 12: Ausgabe der Tripel, bei denen sowohl das Subjekt als auch das Objekt als Entitäten in Yovisto vorkommen. Ein Reduce-Schritt ist nicht notwendig.

Durch die Filterung wurde die Anzahl der Tripel von ursprünglich 11,1 Millionen auf 2,1 Millionen reduziert.

5.3 Distinkte Menge von Ressourcen ermitteln

Auf Basis der Ausgabe von 5.2 und dem *Articles Categories* Datensatz werden anschließend die distinkten Mengen von Entitäten (siehe Abbildung 13), Prädikaten und Kategorien berechnet. Dieser Schritt ist notwendig, um den Import der Daten in das normalisierte Datenbankschema zu erleichtern und zu beschleunigen.

Die Ausgaben ergaben, dass ca. 0,7 Millionen verschiedene Entitäten, 516 unterschiedliche Prädikate und rund 240.000 Kategorien existieren.

<p>comment: <i>key</i>: line number; <i>value</i>: RDF triple</p> <p>procedure MAP(<i>key</i>, <i>value</i>) EMIT(GETSUBJECT(<i>value</i>), <i>null</i>) EMIT(GETOBJECT(<i>value</i>), <i>null</i>)</p>	<p>comment: <i>key</i>: entity; <i>values</i>: nulls</p> <p>procedure REDUCE(<i>key</i>, <i>values</i>) EMIT(<i>key</i>, <i>null</i>)</p>
---	--

Abbildung 13: *Mapper* und *Reducer* zur Berechnung der distinkten Menge von Entitäten.

5.4 Entitäten (Ober-)Kategorien zuordnen

Für die Zuordnung von Kategorien zu den Entitäten, welche für die Auswahl von falschen Antwortmöglichkeiten benötigt werden (siehe Abschnitt 6.3), werden die beiden Datensätze *Articles Categories* und *Categories (Skos)* benötigt. Die Kategorisierung folgt dem *Simple Knowledge Organization System* (SKOS).²⁰ Neben SKOS steht ein Typensystem der DBpedia zu Verfügung, welches jedoch unvollständig und zum Teil fehlerhaft ist. Aus den genannten Gründen haben wir uns für SKOS entschieden, welches zudem eine Vielzahl von Wissensorganisationssystemen (z.B. Thesauri, Klassifikationsschemata und Taxonomien) unterstützt. Im *Articles Categories* Datensatz sind den Entitäten Kategorien direkt zugeordnet, welche zum Teil sehr speziell sind. Ein Problem bei spezifischen Kategorien ist, dass meistens alle Entitäten dieser Kategorie als richtige Antwort in Frage kommen und somit keine falschen Antwortmöglichkeiten ausgewählt werden können. Zum Beispiel sprechen alle Personen aus der 'English Entertainers' Kategorie die Sprache Englisch, so dass keine Person ausgewählt werden kann, die kein Englisch spricht. Die allgemeinere Oberkategorie 'European Actors' ist hier eine bessere Wahl.

Um nun alle Oberkategorien einer Kategorie zu erhalten, haben wir die transitive Hülle über den Kategorien des *Categories (Skos)* Datensatzes gebildet (siehe Abbildung 14). Da einer Oberkategorie wiederum eine Oberkategorie zugeordnet sein kann, muss der MapReduce-Job mit der Ausgabe der vorhergehenden Ausführung wiederholt ausgeführt werden, bis keine neuen Schlussfolgerungen mehr hinzukommen [UKOH09, Urb09]. Anschließend können den Entitäten die Oberkategorien explizit zugeordnet werden.

```

categoriesRelations ← GETCATEGORIESRELATIONS()

comment: key: line number; value: RDF triple
procedure MAP(key, value)
  category ← GETCATEGORY(value)
  broaderCategory ← GETBROADERCATEGORY(value)

  broaderCategories ← GETBROADERCATEGORIES(categoriesRelations, broaderCategory)
  for bc ∈ broaderCategories
    do EMIT(category, bc)

```

Abbildung 14: Bilden der transitiven Hülle über den Kategorien durch iteratives ausführen des MapReduce-Jobs. Ein Reduce-Schritt ist nicht notwendig.

Um darüber hinaus die Generierung von Fragen zu beschleunigen, werden nur Kategorien in die Datenbank importiert, die mindestens 500, aber nicht mehr als 15.000 Entitäten kategorisieren. Aufgrund der Bedingung werden von den rund 240.000 Kategorien, nur

²⁰<http://www.w3.org/2004/02/skos/>

1.250 in die Datenbank geladen. Diese Einschränkungen haben zudem zur Folge, dass sehr allgemeine - oftmals auch falsche - Kategorien ausgeschlossen werden. Zu spezielle Kategorien sind für uns ebenso irrelevant, da für die Generierung von Fragen mehrere Entitäten aus einer Kategorie ausgewählt werden müssen.

5.5 Entitäten gewichten

Die Einteilung der Tripel in den verschiedenen Schwierigkeitsstufen (siehe Abschnitt 6.2) erfolgt anhand der Gewichte von Entitäten, welche in diesem Schritt berechnet werden. Um die Entitäten gewichten zu können, versuchen wir den Bekanntheitsgrad der Entitäten abzuschätzen. Denn umso bekannter die Entitäten einer Frage sind, desto leichter lässt sich diese Frage vermutlich beantworten. Als Grundlage für unseren Ansatz nutzen wir den Wikipedia-Link-Graph (*Pagelinks* Datensatz). Um nun die Popularität einer Entität zu bestimmen, stützen wir uns auf Googles PageRank-Ansatz[BP98]: Je mehr Verlinkungen existieren, desto bekannter ist die Entität. Wir zählen dazu die Anzahl der Links, die auf die zu gewichtene Entität (Wikipedia-Seite) verweisen (siehe Abbildung 15). Anders als beim PageRank-Algorithmus beachten wir jedoch nicht das Gewicht der verlinkenden Seiten.

comment: <i>key</i> : line number; <i>value</i> : RDF triple	comment: <i>key</i> : entity; <i>values</i> : numbers
procedure MAP(<i>key</i> , <i>value</i>)	procedure REDUCE(<i>key</i> , <i>values</i>)
EMIT(GETOBJECT(<i>value</i>), 1)	EMIT(<i>key</i> , GETSUM(<i>values</i>))

Abbildung 15: Anzahl der Links, die auf eine Wikipedia-Seite/Entität verweisen, zählen.

Die drei bekanntesten Entitäten sind *United States* (mit 469.663 Verlinkungen), *England* (164.418) und *France* (155.981).

6 Generierung von Fragen

Nach dem Import der aufbereiteten DBpedia-Daten wird die Generierung der Fragen (siehe Abbildung 16) durchgeführt. Dieser Vorgang erfolgt einmalig, so dass die Fragen sofort zur Verfügung stehen und keine zeitaufwändigen Berechnungen bei der Anfrage mehr durchgeführt werden müssen. Die eigentliche Auswahl und Zusammenstellung der Fragen wird online bei jeder Anfrage ausgeführt. Neben der Zuordnung von Schwierigkeitsstufen ist insbesondere die Auswahl der falschen Antwortmöglichkeiten sehr rechenintensiv. Die komplette Generierung der Fragen wird mit - zum Teil sehr komplexen - SQL-Statements realisiert.

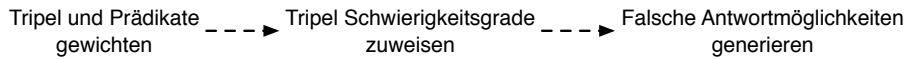


Abbildung 16: Schritte, die bei der Generierung von Fragen ausgeführt werden.

6.1 Das Gewicht von Tripel und Prädikaten berechnen

Basierend auf den Gewichten der Entitäten werden nun die Tripel und Prädikate gewichtet. Anhand dieser Werte werden anschließend, wie im nächsten Kapitel beschrieben, den Tripeln die Schwierigkeitsstufen zugewiesen.

Tripel gewichten Das Gewicht eines Tripels $triple_{weight}$ setzt sich aus dem Gewicht des Subjekts und des Objekts zusammen.

$$triple_{weight} = \frac{44 * subject_{weight}^{triple} + 1 * object_{weight}^{triple}}{45} \quad (1)$$

Da das Subjekt vom Spieler ausgewählt werden muss, wurde dessen Auswirkung auf $triple_{weight}$ erhöht. Dazu haben wir den Einfluss von verschiedenen Koeffizienten untersucht. Der willkürlich gewählte Faktor 44 für $subject_{weight}^{triple}$ stellte sich dabei - subjektiv betrachtet - als bester Wert heraus.

Prädikate gewichten Für die Gewichtung der Prädikate haben wir folgenden Ansatz gewählt: Je bekannter die Subjekte und Objekte eines Prädikates sind, desto wichtiger ist das Prädikat selbst. Dazu werden die Gewichte aller Subjekte $subjects_{weight}^{predicate}$ und Objekte $objects_{weight}^{predicate}$, die mit dem Prädikat in Beziehung stehen, benötigt. Von diesen Gewichten wird jeweils das 5%-winsorisierte Mittel [How07] gebildet. Das Gewicht des Prädikats $weight_{predicate}$ ergibt sich schließlich durch mitteln der berechneten 'winsorisierten' Mittelwerte.

$$weight_{predicate} = \frac{winsor_5(subjects_{weight}^{predicate}) + winsor_5(objects_{weight}^{predicate})}{2} \quad (2)$$

Bei der Mittelwertbildung nach Charles P. Winsor werden die Werte zunächst 'winsorisiert', um den Einfluss von sogenannten 'Ausreißern' zu minimieren. Einer solcher

Ausreißer - über alle Entitäten betrachtet - ist *United States*. Das Gewicht dieser Entität ist fast drei mal so groß wie das zweithöchste Gewicht, der Entität *England*.

$$winsor_{10}(x_1, \dots, x_{10}) = \frac{\overbrace{x_2 + x_2} + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + \overbrace{x_9 + x_9}}{10} \quad (3)$$

Beim 'Winsorisieren' werden die $p\%$ Ausreißer am Anfang und Ende der sortierten Folge durch den nächstgrößeren (bzw. -kleineren) Wert der restlichen Zahlen ersetzt. Die Gleichung 3 zeigt ein Beispiel zur Bildung des 10%-winsorisierten Mittel mit 10 aufsteigend sortierte Zahlen x_1, \dots, x_{10} .

6.2 Tripel den Schwierigkeitsgrad zuweisen

Anhand der im vorhergehenden Schritt berechneten Gewichte werden nun den Tripeln die Schwierigkeitsstufen für die Eins-zu-Eins-Strategie und Eins-zu-N-Strategie zugewiesen.

6.2.1 Eins-zu-Eins-Strategie

Die Zuteilung eines Tripels zu einem Level der Eins-zu-Eins-Strategie ergibt sich ausschließlich aus dem Gewicht des Tripels. So werden die 75 Tripel mit den bekanntesten Subjekt-Objekt-Paaren der Schwierigkeitsstufe 1 zugeordnet, die 75 nächstbesten dem Level 2 usw. (siehe Abbildung 17). Die Anzahl von 75 Tripel pro Level wurde von uns nach eigenem Ermessen gewählt.

```
SELECT id, (row_number() OVER (ORDER BY weight DESC) - 1) / 75 + 1 AS level
FROM triples
WHERE weight IS NOT NULL;
```

Abbildung 17: Zuordnung der Level für die Eins-zu-Eins-Strategie zu den Tripeln.

6.2.2 Eins-zu-N-Strategie

Die Zuordnung der Level für die Eins-zu-N-Strategie erfordert zunächst die Einteilung der Prädikate in Schwierigkeitsstufen analog zu 6.2.1. Für jeden Schwierigkeitsgrad werden jedoch nicht 75, sondern fünf Prädikate ausgewählt. Anschließend werden die 100 besten Prädikat-Objekt-Paare für jedes Prädikat-Level berechnet (siehe Abbildung 18). Die Konstanten wurden auch hier von uns beliebig gewählt. Für jedes der Prädikat-Objekt-Paare wird als Nächstes das Tripel mit dem bekanntesten Subjekt als Repräsentant²¹ für das Paar ermittelt. Die ausgewählten Tripel erhalten letztendlich das Level des zugehörigen Prädikats.

²¹Die Auswahl von falschen Subjekten erfolgt aus dem Pool (siehe Abschnitt 6.3) des Repräsentanten.

```

SELECT rs.p_id, rs.o_id, rs.level
FROM (SELECT p.id AS p_id, o.id AS o_id, p.level AS level,
      row_number() OVER (PARTITION BY p.level
                        ORDER BY (p.weight+o.weight)/2 DESC) AS Rank
FROM triples t
      INNER JOIN predicates p ON (t.predicate_id = p.id)
      INNER JOIN entities o ON (t.object_id = o.id)
WHERE p.weight IS NOT NULL
      AND o.weight IS NOT NULL
GROUP BY p.id, o.id, p.level, p.weight, o.weight) rs
WHERE Rank <= 100;

```

Abbildung 18: Unter Berücksichtigung des Prädikat- und Objekt-Gewichts, die 100 besten Prädikat-Objekt-Paare für jedes Prädikat-Level berechnen.

6.3 Generierung von falschen Antwortmöglichkeiten

Da die Auswahl von falschen Subjekten sehr rechenintensiv ist, wird im Voraus für jedes Tripel ein Pool von falschen Antwortmöglichkeiten berechnet. Bei der Zusammenstellung der Frage werden dann aus den zum Tripel gehörenden Pool, x beliebige Subjekte ausgewählt und mit der/den richtigen Antwort(en) vermischt.

6.3.1 Beste Kategorie des Subjekts ermitteln

Um die Fragen möglichst themenspezifisch zu gestalten, erfolgt die Auswahl von falschen Antwortmöglichkeiten für ein Tripel $\langle s, p, o \rangle$ aus einer Kategorie des Subjekts. Da Entitäten mehreren Kategorien angehören können, muss zunächst die beste Subjekt-Kategorie C_s für jedes Tripel ermittelt werden. Dazu wird als erstes die Kategorie C (mit $s \in C$ und $\langle s_{1\dots x} \subseteq C, p, o \rangle$, $x \geq 1$) ausgewählt, der die meisten Subjekte $s_{1\dots x}$ angehören. Sollte es mehrere solcher Kategorien mit der gleichen Anzahl an Subjekten geben, dann wird die spezifischste Kategorie als C_s ausgewählt (siehe Abbildung 19).

```

SELECT t.id, (SELECT c.id
      FROM entities_categories ec
            INNER JOIN triples t2
                  ON (t2.subject_id = ec.entity_id
                    AND t2.predicate_id = t.predicate_id
                    AND t2.object_id = t.object_id)
            INNER JOIN categories c ON (ec.category_id = c.id)
            INNER JOIN entities_categories ec2
                  ON (ec2.entity_id = t.subject_id
                    AND t2.subject_id = ec2.entity_id
                    AND ec2.category_id = c.id)
GROUP BY c.id, c.frequency
ORDER BY COUNT(*) DESC, c.frequency
LIMIT 1)
FROM triples t;

```

Abbildung 19: SQL-Abfrage zur Auswahl der besten Subjekt-Kategorie für jedes Tripel.

6.3.2 Falsche Antwortmöglichkeiten auswählen

Für jedes Tripel $\langle s, p, o \rangle$ werden nun die falschen Subjekte S_f aus C_s ausgewählt. Hierbei gilt zum einen $s \notin S_f$ und zum anderen $o \neq o_f$ (mit $\langle s_f, p_f, o_f \rangle$, $s_f \in S_f$). Auf die zusätzliche Einschränkung $p \neq p_f$ wurde verzichtet, da aufgrund der 'Open World Assumption' (OWA) die Aussage $\langle s_f, p, o \rangle$ nicht ausgeschlossen werden kann. Durch die Überprüfung der Bedingung $o \neq o_f$ über alle Objekte o_f wird die Wahrscheinlichkeit, dass eine solche Aussage wahr ist, minimiert.

Die zehn falschen Antwortmöglichkeiten pro Tripel werden letztendlich zufällig aus den 25 bekanntesten falschen Subjekten ausgewählt. Somit ist sichergestellt, dass die Antwortmöglichkeiten bei Tripel mit gleicher Subjekt-Kategorie variieren.

7 Auswertung von Spielen

Im Folgenden wird erläutert, wie vom Nutzer abgeschlossene Spiele ausgewertet werden. Diese Auswertung umfasst dabei die folgenden Aspekte:

Punktemodell Das Punktemodell legt fest, wie viele Punkte, Leben und Level²² der Spieler für ein Spiel bekommt oder verliert.

Spielstatistiken Um Rückschlüsse auf die Wichtigkeit der Properties innerhalb der DBpedia zu erlauben, werden Statistiken aus allen Spielergebnissen erstellt.

Benutzerbewertung Dem Benutzer soll die Möglichkeit gegeben werden, Spielergebnisse zu verifizieren und nichtlösbare Spiele zu melden.

7.1 Das Punktemodell

Motivation Die Vergabe von Punkten erlaubt es, die Leistungen verschiedener Spieler zu messen und zu vergleichen. Auf diese Weise kann ein Spieler beispielsweise beurteilen, wie gut er bei *WhoKnows?* im Vergleich zu seinen Freunden abschneidet. Weiterhin wird durch die Vergabe von Punkten eine Highscoreliste und die Freischaltung von Boni ermöglicht. Diese können den Spieler motivieren, das Spiel immer wieder zu spielen, um immer bessere Punktezahlen zu erreichen.

Statische und dynamische Punktemodelle Im Folgenden wird zwischen statischen und dynamischen Punktemodellen unterschieden. Bei einem *statischen Punktemodell* erhält der Benutzer nach dem Spiel eine feste Punktezahl, abhängig von den von ihm gegebenen Antworten. Diese Punktezahl wird nicht mehr verändert, wenn sie einmal gutgeschrieben wurde. Im Gegensatz dazu kann sich bei einem *dynamischen Punktemodell* nachträglich noch die vergebene Punktezahl ändern. Diese Variante ist vor allem dann sinnvoll, wenn nach Spielende noch neue Erkenntnisse die Bewertung des Spiels beeinflussen können. Sind beispielsweise bei einem Quiz-Spiel die korrekten Antworten nicht bekannt, so kann ein Spieler Punkte verdienen, wenn andere Teilnehmer später die selbe Antwort wählen wie er.

Im Fall von *WhoKnows?* existiert durch die DBpedia-Daten ein Grundwissen, das zunächst als korrekt angenommen werden kann. Aus diesem Grund kann sofort nach einem Spiel entschieden werden, welche Antworten korrekt gegeben wurden und welche nicht, so dass ein statisches Punktemodell verwendet werden kann. Dieses bietet den Vorteil, dass es für den Spieler leichter durchschaubar ist, da Punktwerte nachträglich nicht mehr geändert werden.

Punktemodell für Eins-zu-Eins-Spiele Tabelle 1 zeigt das Punktemodell für Eins-zu-Eins-Spiele. Je nach Spielausgang (der Spieler wählt die richtige, falsche Antwort oder die Zeit läuft ab) ändert sich die Zahl der Punkte und Leben sowie das aktuelle

²²*Leben* gibt die Anzahl der verbleibenden Versuche für den Spieler an, *Level* bestimmt den Schwierigkeitsgrad der gestellten Fragen

Level des Spielers. Der Spieler erhält dabei für eine korrekte Antwort Pluspunkte, für eine falsche oder gar keine Antwort Minuspunkte. Durch den höheren Verlust bei einer nichtgegebenen Antwort im Vergleich zu einer falschen Antwort wird der Spieler motiviert sich vor Ablauf der Zeit für eine Antwort zu entscheiden.

	Punkte	Level	Leben
Zeit abgelaufen	-30	-1	-1
Antwort richtig	+20 (+ Zeitbonus)	+1	0
Antwort falsch	-20	-1	-1

Tabelle 1: Punktemodell für Eins-zu-Eins-Spiele

Punktemodell für Eins-zu-N-Spiele Tabelle 2 zeigt die Punktevergabe für Eins-zu-N-Spiele. In diesem Fall ist das System komplexer, da der Benutzer auch *teilweise* richtige Antworten geben kann.

Bei der Bewertung eines Spiels werden daher alle Antwortmöglichkeiten einzeln betrachtet und jeweils mit Hilfe der folgenden vier Fälle bewertet:

- die Antwort ist richtig und wurde vom Spieler ausgewählt: +10 Punkte
- die Antwort ist falsch und wurde vom Spieler ausgewählt: -10 Punkte
- die Antwort ist richtig und wurde vom Spieler nicht ausgewählt: -5 Punkte
- die Antwort ist falsch und wurde vom Spieler nicht ausgewählt: +5 Punkte

Die Summe dieser Werte für alle Antwortmöglichkeiten ist der Punktegewinn des Spielers. Zusätzlich wird ein Zeitbonus von einem Punkt pro verbleibender Zeiteinheit vergeben, falls der Spieler ausschließlich korrekte Antworten vergeben hat. Waren zumindest einige seiner Antworten korrekt, erhält er einen halben Punkt pro Zeiteinheit.

	Punkte	Level	Leben
Zeit abgelaufen	-30	-1	-1
Antwort richtig	(abhängig von Antworten)	wenn alle richtig: +1 sonst: 0	wenn alle richtig: 0 sonst: -1
Antwort falsch	(abhängig von Antworten)	-1	-1

Tabelle 2: Punktemodell für Eins-zu-N-Spiele

7.2 Erstellen von Spielstatistiken

Ein wichtiges Ziel des Spiels *WhoKnows?* ist es, die Prädikate aus der DBedia für die Verwendung in der explorativen Suche von Yovisto zu evaluieren. Dazu muss aus den Ergebnissen der Spieler ermittelt werden, welche Prädikate als besonders wichtig wahrgenommen werden.

Der Auswertung der Spielergebnisse liegt dabei folgende Annahme zugrunde: Je öfter ein Prädikat mit einem Objekt durch verschiedene Spieler seinem zugehörigen Subjekt zugeordnet werden kann, desto wichtiger ist das Prädikat-Objekt-Paar für die Beschreibung des Subjekts. Wird beispielsweise das Property *dbpedia-owl:knownFor* in vielen Fällen mit unterschiedlichen Subjekten und Objekten korrekt zugeordnet, so wird es als wichtiges Prädikat betrachtet. Prädikate, die nur selten korrekt zugeordnet werden, werden hingegen als unwichtig gewertet.

Bei der Generierung eines Spiels wird festgehalten, welche Tripel aus den DBpedia-Datensätzen es enthält. Bei der Auswertung kann anschließend verglichen werden, welche der Tripel durch den Benutzer korrekt zugeordnet wurden. *WhoKnows?* speichert nun für jedes gespielte Tripel die Zahl, wie oft es in Spielen vorgekommen ist und die Zahl, wie oft es davon korrekt zusammengesetzt wurde. Das Verhältnis dieser Werte für alle Tripel mit dem selben Prädikat gibt Aufschluss über die Wichtigkeit dieses Prädikats.

Eine Schwierigkeit dieser Evaluation ist, dass für ein gutes Spielerlebnis eine Vorauswahl an Fragen getroffen werden muss. Diese muss beispielsweise für einen steigenden Schwierigkeitsgrad der Fragen im Verlauf des Spiels sorgen. Durch diese Maßnahmen wird jedoch auch Einfluss auf die Auswertung der Spielerantworten genommen (beispielsweise werden leichtere Fragen öfter gespielt, da sie mit hoher Wahrscheinlichkeit gleich zu Anfang verwendet werden). In Abschnitt 9.1 wird diese Problematik näher erläutert.

7.3 Bewertung des Spiels durch den Benutzer

Im Folgenden wird erläutert, welche Möglichkeiten *WhoKnows?* dem Spieler bietet, das Ergebnis eines Spiels zu kontrollieren und gegebenenfalls zu beanstanden.

Diese Funktionen sind notwendig, da die DBpedia-Daten weder vollständig noch in allen Fällen korrekt sind. So können beispielsweise Fragen generiert werden, die mehrere korrekte Antworten haben, von denen aber nur eine als richtig erkannt wird (vgl. Abschnitt 9.1).

7.3.1 Anzeigen von Informationen zu Entities

Wenn der Spieler eine Frage beantwortet, wird ihm im Anschluss das richtige Ergebnis präsentiert. Dieses ist jedoch zunächst schwer nachvollziehbar, wenn sich der Spieler im jeweiligen Fachgebiet nicht auskennt. Aus diesem Grund bietet *WhoKnows?* dem Spieler weitere Hintergrundinformationen. Diese kann er benutzen, um einerseits das angezeigte Ergebnis zu verifizieren (zum Beispiel kann er überprüfen, ob das vom Spiel angezeigte Ergebnis tatsächlich so in der Wikipedia steht) und andererseits kann er sich bei Interesse näher über einen Begriff informieren.

Um zusätzliche Informationen abzurufen bietet das Spiel drei Möglichkeiten (siehe Abbildung 20):

- Das Anklicken des Wikipedia-Icons neben einem Entity öffnet ein Fenster mit allgemeinen Informationen zu dem ausgewählten Objekt.
- Innerhalb des Informationsfensters kann der Nutzer über den Button *More* direkt zum vollständigen Wikipediaartikel wechseln.

- Zusätzlich existiert neben jedem Entity ein DBpedia-Icon. Dieses öffnet die DBpedia-Seite, auf der alle Properties des Entities aufgelistet werden.

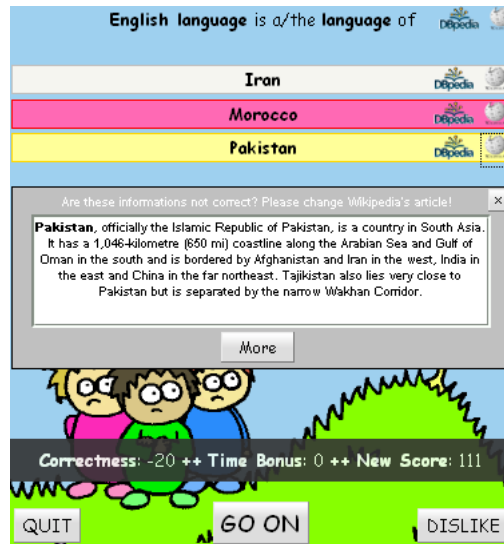


Abbildung 20: Screenshot aus *WhoKnows?*: Möglichkeiten zum Abrufen von detaillierten Informationen über Entities.

Für die angezeigten allgemeinen Informationen über Entities wird dabei ein Teil des jeweiligen *Wikipedia-Abstracts* verwendet. Um auf diesen zuzugreifen existieren verschiedene Möglichkeiten:

1. Die Daten werden bei jedem Zugriff von der offiziellen Wikipediaseite heruntergeladen und aus der HTML-Seite extrahiert. Dieses Vorgehen hat den Vorteil, dass stets die aktuellsten Daten zur Verfügung stehen. Allerdings ist die Extraktion nicht einfach, da nicht alle Wikipediaseiten genau dem selben Schema folgen. Weiterhin müssen bei vielen Zugriffen auf das Spiel auch viele, vergleichsweise langsame, Anfragen an die Wikipedia-Webserver gestellt werden.
2. Die Daten werden am DBpedia SPARQL-Endpoint²³ abgefragt. Dadurch kann die Anfrage deklarativ gestellt werden und die Antwort lässt sich ohne großen Aufwand mit Hilfe einer SPARQL-Bibliothek verwenden. Als Nachteil bleibt jedoch, dass für jede Informationsanfrage ein Zugriff auf den DBpedia SPARQL-Endpoint erfolgen muss.
3. Eine weitere Möglichkeit ist es, alle Wikipedia-Abstracts lokal vorzuhalten. Dadurch muss keine zusätzliche Netzwerkverbindung aufgebaut werden und das Spiel ist nicht von der DBpedia als externe Quelle abhängig. Allerdings muss dafür eine sehr hohe Datenmenge²⁴ lokal gespeichert werden. Weiterhin müssen die Daten

²³<http://dbpedia.org/sparql>

²⁴Die unkomprimierten Abstract-Triples sind unkomprimiert ca. 1,2 GB groß

in regelmäßigen Abständen aktualisiert werden, da sie sonst nicht mehr zu dem verlinkten Wikipediaartikel passen.

Das Spiel *WhoKnows?* implementiert die zweite Variante, da diese einen guten Kompromiss zwischen einfacher Abfragbarkeit, Aktualität und zu speichernder Datenmenge bietet.

7.3.2 Die Dislike-Funktion

Stellt der Nutzer fest, dass eine ihm gestellte Frage nicht eindeutig beantwortbar ist, so benötigt er eine Möglichkeit diese zu melden. Dies geschieht über den *Dislike-Button* im Ergebnisbildschirm.

Markiert ein Nutzer auf diese Weise das aktuelle Spiel, werden alle Daten, die für die Rekonstruktion des kompletten Spiels notwendig sind²⁵, in separaten Tabellen gesichert. Diese Daten bieten die Möglichkeit, einzelne oder systematische Datenfehler zu finden und gegebenenfalls zu korrigieren. Sind beispielsweise die Angaben in einem Wikipediaartikel falsch und werden deshalb nicht-beantwortbare Fragen generiert, so kann der Spieler dies über den Dislike-Button mitteilen. Auf diese Weise kann im Laufe der Zeit die Datenqualität verbessert werden und für den Spieler steigt der Spielspaß, da er nur mit lösbaren Fragen konfrontiert wird.

²⁵Diese beinhalten den Zeitpunkt des Spiels, den Spielernamen, die Spielvariante, sowie alle Angaben zur korrekten Lösung und den angebotenen Antwortmöglichkeiten.

8 Soziale Netzwerke

Soziale Netzwerke erfreuen sich immer größerer Beliebtheit, nicht zuletzt sind beispielsweise Facebook²⁶, Twitter²⁷, MySpace²⁸ oder LinkedIn²⁹ unter den Top 25 der am meisten besuchten Internetseiten zu finden.³⁰ Neben den typischen Funktionen wie Nachrichten versenden, Fotos teilen usw. bieten die meisten Netzwerke Möglichkeiten zur Entwicklung von Applikationen, welche dann innerhalb dieser Netzwerke laufen. Besonders bei Spielen - so genannten Social Games - hat sich mittlerweile ein riesiger Markt entwickelt, auf den sich sogar schon Firmen spezialisiert haben. Grundsätzlich sind sämtliche Spiele kostenlos, allerdings ergeben sich durch Werbeeinhalte oder den Erwerb einer virtuellen Währung in Verbindung mit den hohen Nutzerzahlen profitable Geschäftsmodelle [Che09]. In einer Umfrage wurde sogar festgestellt, dass sich fast die Hälfte der Mitglieder sozialer Netzwerke überwiegend wegen den Social Games anmeldet [Ing10]. Für *WhoKnows?* bedeuten soziale Netzwerke demnach beste Voraussetzungen eine große Nutzerbasis für das Spiel zu gewinnen. Darüber hinaus wird eine Benutzerverwaltung mitgeliefert, auf die *WhoKnows?* angewiesen ist, da das Spiel selbst eine solche nicht mitbringt.

8.1 Facebook als soziales Netzwerk

Facebook wurde im Februar 2004 ursprünglich als Plattform für Harvard Studenten online gestellt, wurde jedoch nach und nach erst für weitere Universitäten in den USA, dann für Nicht-Studenten und bald auch für andere Länder freigeschaltet [Wik10b]. Heute zählt Facebook mehr als 500 Millionen Mitglieder, wurde in mehr als 70 Sprachen übersetzt [Rot10] und kann daher als das am weitesten verbreitete soziale Netzwerk der Welt betrachtet werden³¹. Seit Mai 2007 existiert die 'Facebook Developer Platform'³², welche einige APIs zur Verfügung stellt, die es Dritten erlaubt Anwendungen zu entwickeln, die Daten aus Facebook lesen und nach Facebook schreiben. Eine API für mobile Facebook-Anwendungen steht ebenfalls bereit - bei mehr als 150 Millionen aktiven Nutzern, die Facebook über mobile Endgeräte besuchen [Rot10], ist das sicherlich kein Wunder. Facebook stellt eine Art Vorreiter für Social Games dar und ist mit Abstand die beliebteste Plattform für solche Spiele [Ing10]. Mittlerweile existieren über 550.000 aktive Anwendungen, wobei 70% aller Mitglieder zumindest eine dieser Applikationen jeden Monat nutzen [Rot10]. Selbst wenn nur ein Bruchteil dieser Mitglieder *WhoKnows?* spielt, weil sie über Mundpropaganda davon erfahren haben oder doch eher zufällig auf das Spiel gestoßen sind, besteht immer noch ein riesiger Markt an potenziellen Spielern, die im Laufe der Zeit genug statistische Daten für den eigentlichen Zweck des Spiels anhäufen können sollten.

²⁶<http://www.facebook.com>

²⁷<http://twitter.com>

²⁸<http://www.myspace.com>

²⁹<http://www.linkedin.com>

³⁰<http://www.alexa.com/topsites/global>

³¹http://en.wikipedia.org/wiki/List_of_social_networking_websites

³²<http://developer.facebook.com>

8.2 Einbettung des Spiels in Facebook

Die Integration einer Anwendung in Facebook erfolgt prinzipiell durch einen iFrame³³, welcher auf der von Facebook gehosteten Webseite der Applikation³⁴ liegt. Der iFrame referenziert dann auf eine vom Entwickler definierte URI, wo die Applikation tatsächlich läuft [Fac10a]. *WhoKnows?* macht dabei von der Graph API [Fac10c] Gebrauch, über die das Spiel Nutzerdaten erfragt, wobei der Spieler allerdings vor der Verwendung der Applikation erst sein Einverständnis geben muss, dass er seine Daten überhaupt zur Verfügung stellt. Bevor eine Anwendung auf Daten eines Nutzers zugreifen kann, muss sie zunächst das *access token* ermitteln. Die Abbildung 21 zeigt die dazu notwendigen Schritte [Fac10b].



Abbildung 21: Ermittlung des *access tokens*

Mit Hilfe des *access tokens* können dann durch Aufruf einer URI bestimmte Daten erfragt werden, die als JSON-Zeichenkette zurückgegeben werden. Im Fall von *WhoKnows?* werden lediglich allgemeine Daten wie Name und Profillink sowie die URI des Profilbildes erfragt (siehe Tabelle 3).

URI	Ausgabe
<code>https://graph.facebook.com/me?access_token= <access_token></code>	{“id“ : “100001368090534“, “name“ : “Jim Panse“, “first_name“ : “Jim“, “last_name“ : “Panse“, “link“ : http://www.facebook.com/profile.php?id=100001368090534, “timezone“ : 2, “locale“ : “en_GB“, “verified“ : true, “updated_time“ : “2010-07-06T08:43:25+0000“ }
<code>https://graph.facebook.com/me/picture?access_token= <access_token></code>	Umleitung zur eigentlichen Bildquelle

Tabelle 3: Verwendung der Graph API

Das *me* vor dem Querystring steht dabei für den angemeldeten Benutzer selbst, man könnte dort allerdings genauso gut die Facebook Benutzer ID einsetzen. Die Daten werden dann in die eigens für das Spiel bestimmte Datenbank übertragen, um entsprechende Referenzen zu Highscoreinträgen usw. zu schaffen.

³³eine Webseite, die in eine andere Webseite eingebettet ist

³⁴http://apps.facebook.com/whoknows_/

Neben der Graph API wird darüber hinaus noch das JavaScript SDK³⁵ verwendet, das es unter anderem erlaubt, einen Dialog anzuzeigen, der einen Eintrag auf der eigenen Facebookwand ermöglicht, welcher dann für Freunde sichtbar ist (siehe Abbildung 22). Wenn jener Spieler eine besonders hohe Punktzahl erreicht hat und dadurch möglicherweise einen Freund übertroffen hat, wäre es für diesen eine zusätzliche Motivation, das Spiel erneut zu spielen.



Abbildung 22: Dialog für einen Eintrag auf der Wand

³⁵<http://developers.facebook.com/docs/reference/javascript>

9 Projekt Evaluation

An dieser Stelle soll eine kleine Evaluation des Projektes vorgenommen werden. Zunächst wird auf Schwierigkeiten eingegangen, die sich im Laufe des Projektes ergeben haben, woraufhin erste statistische Daten, die sich auf den eigentlichen Zweck des Spiels beziehen, ausgewertet werden. Sämtliche Untersuchungen basieren auf einem Datenbestand, der in 117 Spielen angehäuft wurde. Dabei wurden 882 distinkte Tripel verwendet, welche insgesamt 3.040 Mal gespielt wurden.

9.1 Probleme und Anmerkungen

Beeinflussung der Statistiken durch Gewichtung Wie in Abschnitt 5.5 bereits geschildert, werden Entitäten anhand der Anzahl der Verlinkungen auf diese Entität gewichtet. Daraus ergeben sich wiederum die Gewichte für Tripel und Prädikate. Aufgrund der Tatsachen, dass diese Gewichte den Schwierigkeitsgrad eines Tripels bestimmen und es relativ schwer ist, hohe Level mit entsprechend hohem Schwierigkeitsgrad zu erreichen, folgen daraus logische Konsequenzen:

1. Tripel, bei denen das Subjekt ein relativ geringes Gewicht hat, werden sehr selten, d.h. von nur sehr guten Spielern, gespielt.
2. Tripel, bei denen das Subjekt ein hohes Gewicht hat, werden je nach Gewicht des Objektes bzw. des Prädikates ausreichend häufig für die Statistik gespielt.

Aus der ersten Schlussfolgerung ergibt sich, dass für die Entitäten, welche als Subjekt in den Tripeln vorhanden sind, wahrscheinlich nie eine Aussage über die Wichtigkeit der einzelnen Properties der Entität gemacht werden kann, da einfach nicht genügend Daten vorhanden sind. Um dem entgegenzuwirken könnten in unteren Levels Fragen, die aus selten gespielten Tripeln erstellt werden, eingestreut werden. Aus der zweiten Konsequenz lässt sich schließen, dass für diese Entitäten genug Daten angehäuft werden, um eine Aussage über das Ranking der Properties zu machen. Irgendwann sind diese Entitäten jedoch so häufig gespielt worden, dass die fortführende Sammlung an Statistiken für diese Entitäten nicht mehr notwendig ist. Für diesen Fall wäre es mit Sicherheit sinnvoll, die entsprechenden Tripel für Fragestellungen auszuschließen. Andere selten gespielte Tripel, die normalerweise in höheren Level gespielt werden, könnten diese ausgeschlossenen Tripel ersetzen, so dass nach und nach für alle Entitäten genug statistische Daten angehäuft werden. Logischer Nachteil dieses Ansatzes wäre, dass das Spiel im Laufe der Zeit selbst in unteren Levels immer schwieriger wird. Bis dato wurden gerade mal 538 von 686.220 Entitäten sowie 882 von 2.125.116 Tripeln gespielt - und das, obwohl immerhin schon Level 22 als Maximum im Spiel erreicht wurde. Auch wenn demnach wenig Daten in der Breite angehäuft werden, hat sich gezeigt, dass die Einführung der Gewichtungen unerlässlich war, da das Spiel zuvor zu schwierig und daher fast unspielbar war.

Hierarchische Probleme Viele der seltsam erscheinenden Fragen im Spiel sind auf das gleiche Problem zurückzuführen. Ein Beispiel: Das Prädikat für die Frage lautet *liegtIn* und das Objekt ist *Deutschland*. Als Antwortmöglichkeiten sind *Berlin*, welches die

angeblich richtige Antwort darstellt, sowie *München* vorgegeben. Die Entität *München* hat für das Prädikat *liegtIn* lediglich das Objekt *Bayern* in der DBpedia zu stehen, was allerdings bekannterweise ebenfalls in *Deutschland* liegt, weshalb *München* eigentlich auch richtig sein müsste. Um solche seltsamen Fragen zu verhindern, kann man die transitive Hüllen der Objekte des *liegtIn*-Prädikats bilden. Dabei muss natürlich die Hierarchiebeziehung, dass *Bayern* nämlich in *Deutschland* liegt und nicht umgekehrt, bekannt sein. Existiert nun also eine Entität *Augsburg*, die beim Prädikat *liegtIn* die Objekte *Bayern* und *Deutschland* zu stehen hat, kann man das Objekt *Deutschland* problemlos beim *liegtIn*-Prädikat von *München* hinzufügen. Ist dies geschehen, kann konsequenterweise *München* nicht mehr als falsche Antwortmöglichkeit bei dieser Frage auftauchen.

Weitere Ursachen für seltsame Fragen Eine häufige Ursache für mangelhafte Datenqualität in der DBpedia ist die fehlerhafte Anwendung der MediaWiki Syntax innerhalb der Infoboxen auf Wikipediaseiten und die daraus resultierende fehlerhafte Extraktion dieser Infoboxen, wie sie in [ABK⁺07] beschrieben ist. Die Abbildung 23 und 24 zeigen, dass bei der Extraktion der Infoboxen immer konsequent die Inhalte der WikiLinks - ein Link, der auf eine andere Wikipediaseite verlinkt - verwendet werden. Da an dieser Stelle *De jure* und *De facto* als WikiLinks gekennzeichnet wurden, treten sie letztendlich auch wieder als Objektentitäten in der DBpedia auf. Möglicherweise sollte man den Extraktor in diesem Zusammenhang etwas flexibler gestalten.



Abbildung 23: Connecticut (Wikipedia)

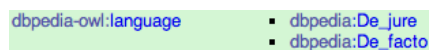


Abbildung 24: Connecticut (DBpedia)

Die Abbildung 25 zeigt, dass für Alaska in der Wikipedia die Sprachen *English* und *Native North American* nicht als WikiLinks gekennzeichnet wurden und demzufolge in der DBpedia als Objektentitäten zum Prädikat *language* gar nicht auftauchen (siehe Abbildung 26).

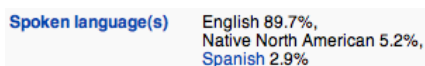


Abbildung 25: Alaska (Wikipedia)

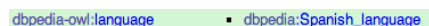


Abbildung 26: Alaska (DBpedia)

Beide genannten Beispiele haben zur Folge, dass möglicherweise diese Entitäten irrtümlich als falsche Antwortmöglichkeiten vorgegeben werden, wenn das Prädikat der Frage *language* und das Objekt der Frage *English* lautet.

Darüber hinaus existieren noch viele seltsame Tripel in der DBpedia, die ihren Ursprung allerdings in der Wikipedia haben:

- Wales | anthem | English language
- Pop music | instrument | Singing
- ...

Die Property, die in Abbildung 27 zu sehen ist, befindet sich tatsächlich in dieser Form in der Wikipedia. Jedoch zumindest von solchen Datenqualitätsproblemen ist *WhoKnows?* nicht betroffen, da lediglich Tripel aus der DBpedia betrachtet wurden, bei denen sowohl das Subjekt als auch das Objekt Yovisto-Entitäten sind.

dbpprop:boxWidth ■ 290px

Abbildung 27: Albert Einstein (DBpedia)

9.2 Erste statistische Daten aus der Nutzung

Der Zweck des Spiels besteht bekanntlich darin, für jede Yovisto-Entität alle relevanten Properties entsprechend ihrer Wichtigkeit zu ordnen. Wenn eine Aussage darüber getroffen werden soll, ist das Verhältnis zwischen korrekt erkannten und insgesamt gespielten Objekteigenschaften entscheidend - wie schon in Abschnitt 2.3 beschrieben.

Tabelle 4 zeigt die Statistiken zur Entität *England*, von der tatsächlich schon alle relevanten Properties gespielt wurden. Die Daten sind absteigend nach dem schon angesprochenen relativen Verhältnis korrekter Lösungen sortiert, so dass dies auch gleichzeitig das durch *WhoKnows?* ermittelte Ranking der Properties wäre. Sofern ein Mensch selbst ein Ranking nach seinem subjektiven Empfinden vornehmen würde, würde er vermutlich die Prädikate *language* und *ethnicGroup* weiter unten und das Prädikat *anthem* weiter oben einsortieren. In Bezug auf das Prädikat *language* besteht die einfache Erklärung darin, dass jeder Spieler natürlich die Verbindung des Objektes *English language* zu dem Subjekt *England* herzustellen weiß. In einem von einem Menschen vorgenommenen Ranking würde dieser Property nicht so eine große Relevanz zugesprochen werden, da noch viele andere Länder existieren, in denen Englisch gesprochen wird. Darüber hinaus wurden einige der Properties bislang zu selten gespielt, so dass sie möglicherweise zu hoch rangieren, da die Spieler mit ihrer Antwort zufällig richtig lagen, oder zu niedrig rangieren, weil sich womöglich der eine oder andere Spieler verklickt hat.

Prädikat	Objekt	korrekt	gespielt	relativ
language	English language	12	12	1,0
leaderName	Gordon Brown	2	2	1,0
capital	London	14	15	0,933
governmentType	Constitutional monarchy	11	13	0,846
ethnicGroup	White people	2	3	0,666
leaderName	Elizabeth II of the United Kingdom	3	5	0,6
regionalLanguage	Cornish language	8	15	0,533
anthem	God Save the Queen	1	2	0,5

Tabelle 4: Statistiken zur Entität *England*

Im Grunde genommen lässt sich trotz des geringen Datenbestandes erkennen, dass dieses Spielkonzept funktioniert - beruhend auf der Prämisse, dass für eine bekannte Property einer Entität auch gleichzeitig gilt, dass sie eine wichtige Property für die Entität darstellt.

Der *Dislike-Button* wurde erwartungsgemäß auch schon des Öfteren betätigt. Auffällig ist dabei, dass sich viele Datensätze auf das Prädikat *language* beziehen, für das schon in Abschnitt 9.1 ein paar Beispiele genannt wurden. Interessanterweise gibt es selbst in diesem geringen Datenbestand ein paar schöne Beispiele, bei denen der Aufbau von Hierarchien Abhilfe schaffen könnte:

- richtige Lösung: Country music | instrument | Guitar
- weitere Antwortmöglichkeit: Rock and roll | instrument | Electric guitar
- richtige Lösung: West Virginia | instrument | English
- weitere Antwortmöglichkeit: Lake Taylor High School | instrument | American English

Die Umsetzung dieser Funktionalität hat sich als sinnvoll erwiesen, da auf diese Weise schnell seltsame Fragen protokolliert und gegebenenfalls Datenqualitätsprobleme in DBpedia nachvollzogen werden können. Darüber hinaus wäre es mit Sicherheit sinnvoll, diese seltsame Fragen vom Spiel auszuschließen, sofern der *Dislike-Button* entsprechend häufig gedrückt wurde.

9.3 Konklusion

Wie schon im vorherigen Abschnitt 9.2 angesprochen, funktioniert das Spielkonzept unter der Annahme, dass eine bekannte Property auch gleichbedeutend eine wichtige Property darstellt. Dennoch bestehen im Spiel noch Probleme, die sich einerseits auf die Fragenauswahl im Spiel (zu schwierig und wenig abwechslungsreich) sowie andererseits auf die Datenqualität in der Wikipedia und DBpedia beziehen. Zumindest ist der *Dislike-Button* in dieser Hinsicht eine gute Maßnahme, um konkrete Beispiele für die angesprochenen Schwierigkeiten festzuhalten.

10 Future Work

Das grundlegende Spielkonzept wurde zum jetzigen Zeitpunkt umgesetzt, ermöglicht jedoch die Implementierung des Spiels noch Möglichkeiten für Erweiterungen. Einerseits kann das Spiel mehr Menschen zugänglich gemacht werden, andererseits betrifft dies Erweiterungen, die dem Spielspaß zuträglich sind und damit auch die Motivation steigern das Spiel noch häufiger zu spielen. Beide Betrachtungsweisen haben ein Mehr an gesammelten statistischen Daten zur Folge, was eine deutlichere Aussage bezüglich des Rankings von Properties einer Entität möglich macht. Im Folgenden sollen nun denkbare zukünftige Aufgaben näher erläutert werden.

Bislang wurde das Spiel für das soziale Netzwerk Facebook umgesetzt, kann jedoch aufgrund seiner Architektur **problemlos in andere Netzwerke integriert** werden. Entscheidend ist dabei, dass dieses soziale Netzwerk eine Plattform für die Veröffentlichung solcher Spiele bietet. Zudem muss natürlich eine Benutzeridentifikation über einen Login gegeben sein, da das Spiel selbst über eine solche Funktionalität nicht verfügt. Die Integration des Spiels in MySpace und das dem Facebook sehr ähnlichen studiVZ³⁶ wäre schon allein wegen der hohen Nutzerzahlen (MySpace: 67 Mio; studiVZ: 4.1 Mio³⁷) sehr sinnvoll.

Weiterhin ist die **Einführung anderer Spielvarianten** denkbar. Während bislang nur zu einem Prädikat-Objekt-Paar mehrere Subjekte als mögliche Lösungen vorgegeben wurden, könnten auch mehrere Objekte aufgeführt sein, die dann jeweils zu einem passenden Subjekt zugeordnet werden müssen (M-zu-N-Variante). Bei dieser Variante beziehen sich alle Subjekt-Objekt-Paare auf das gleiche Prädikat. Vorstellbar wäre auch, dass mehrere Prädikat-Objekt-Paare jeweils zu einem passenden Subjekt zugeordnet werden müssen. Dabei sollte allerdings gewährleistet sein, dass alle Subjekte, die als Lösungsvarianten vorgeschlagen werden, alle verwendeten Prädikate mit logischerweise unterschiedlichen Objekten besitzen, um Zweideutigkeiten zu vermeiden. Anderenfalls kann es passieren, dass ein Prädikat-Objekt-Paar zwei verschiedenen Subjekten zugeordnet werden könnte, nur weil bei einem Subjekt jenes Prädikat-Objekt-Paar in der DBpedia gar nicht existiert, in Wirklichkeit jedoch dem Prädikat-Objekt-Paar des anderen Subjektes gleicht.

Darüber hinaus wären neben **Spieltypen** wie 'Hangman' oder 'Formula' noch weitere realisierbar. Beispielsweise könnten Antwortmöglichkeiten auf dem Bildschirm herunterfallen, wobei die jeweils richtigen weggeklickt werden müssen - analog zu bereits existierenden Spielen wie beispielsweise Moorhuhn³⁸. Für die M-zu-N-Variante wären bewegliche Antwortmöglichkeiten denkbar, die per Drag & Drop den richtigen Properties zugeordnet werden sollen. Solche Spieltypen könnten erst ab einem bestimmten Level, das der Spieler je erreicht hat, freigeschaltet werden, wodurch die Motivation nochmals gesteigert wird.

Zudem wäre ein **Bonussystem** denkbar. Boni könnten vergeben werden, wenn der Spieler eine bestimmte Anzahl an Fragen in Folge richtig beantwortet oder eine bestimmte Punktzahl erreicht. Gleichmaßen könnten für diese Spielereignisse **Medaillen** vergeben werden, die der Spieler dann in seiner persönlichen Vitrine begutachten kann. Sämtliche

³⁶<http://www.studivz.de>

³⁷<http://www.google.com/adplanner>

³⁸<http://www.moorhuhn.de>

erlangte Boni könnten dann jederzeit innerhalb eines Spiels eingesetzt werden. Ein solcher Bonus könnte bedeuten, dass der Spieler mehr Zeit für eine Spielrunde besitzt oder dass falsche Antwortmöglichkeiten ausgeblendet werden.

Wie bereits in Abschnitt 5.5 erwähnt, wird der **Google PageRank-Algorithmus** als Basis für die Gewichtung der Entitäten verwendet, wobei allerdings die Gewichte der Entitäten, welche auf die zu gewichtene Entität verweisen, bislang unbeachtet bleiben. Die Einbeziehung dieser Gewichte hat möglicherweise zur Folge, dass Entitäten, die ein hohes Gewicht haben und in eher unteren Levels auftauchen, thematisch gesehen verschiedener sind und nicht nur Länder, Regierungsformen usw. bezeichnen. Für das Spiel würde dies ein wenig mehr Abwechslung in Bezug auf die Themen der Fragen bedeuten.

Literatur

- [ABK⁺07] AUER, Sören ; BIZER, Christian ; KOBILAROV, Georgi ; LEHMANN, Jens ; CYGANIAK, Richard ; IVES, Zachary: DBpedia: A Nucleus for a Web of Open Data. In: *ISWC'07/ASWC'07: Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*. Berlin, Heidelberg : Springer, 2007. – ISBN 3-540-76297-3, 978-3-540-76297-3, S. 722-735
- [AD04] AHN, Luis von ; DABBISH, Laura: Labeling Images with a Computer Game. In: *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM, 2004. – ISBN 1-58113-702-8, S. 319-326
- [AGK⁺06] AHN, Luis von ; GINOSAR, Shiry ; KEDIA, Mihir ; LIU, Ruoran ; BLUM, Manuel: Improving Accessibility of the Web with a Computer Game. In: *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*. New York, NY, USA : ACM, 2006. – ISBN 1-59593-372-7, S. 79-82
- [AKB06] AHN, Luis von ; KEDIA, Mihir ; BLUM, Manuel: Verbosity: A Game for Collecting Common-Sense Facts. In: *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*. New York, NY, USA : ACM, 2006. – ISBN 1-59593-372-7, S. 75-78
- [ALB06] AHN, Luis von ; LIU, Ruoran ; BLUM, Manuel: Peekaboom: A Game for Locating Objects in Images. In: *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*. New York, NY, USA : ACM, 2006. – ISBN 1-59593-372-7, S. 55-64
- [BP98] BRIN, Sergey ; PAGE, Lawrence: The Anatomy of a Large-Scale Hypertextual Web Search Engine. In: *Comput. Netw. ISDN Syst.* 30 (1998), Nr. 1-7, S. 107-117. – ISSN 0169-7552
- [CD88] COULOURIS, George F. ; DOLLIMORE, Jean: *Distributed systems: concepts and design*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1988
- [Che09] CHEN, Sande: *The Social Network Game Boom*. http://www.gamasutra.com/view/feature/4009/the_social_network_game_boom.php?print=1. Version: April 2009
- [DG04] DEAN, Jeffrey ; GHEMAWAT, Sanjay: MapReduce: Simplified Data Processing on Large Clusters. In: *OSDI'04: Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation*. Berkeley, CA, USA : USENIX Association, 2004, S. 10-10
- [Döl10] DÖLLNER, Jürgen: *Lecture Visualisation*. 2010

- [Fac10a] FACEBOOK: *Apps on Facebook.com*. <http://developers.facebook.com/docs/guides/canvas>. Version: August 2010
- [Fac10b] FACEBOOK: *Authentication*. <http://developers.facebook.com/docs/authentication>. Version: August 2010
- [Fac10c] FACEBOOK: *Graph API*. <http://developers.facebook.com/docs/api>. Version: August 2010
- [How07] *Kapitel A Return to Trimming*. In: HOWELL, David C.: *Fundamental Statistics for the Behavioral Sciences*. Cengage Learning Servi, 2007, S. 91–93
- [Ing10] INGRAM, Mathew: *Average Social Gamer Is a 43-Year-Old Woman*. <http://gigaom.com/2010/02/17/average-social-gamer-is-a-43-year-old-woman>. Version: Februar 2010
- [Jen10] JENS, Sonja: *Die Weisheit der Vielen - Problemfelder und Schlüsselkriterien*. <http://www.digitale-unternehmung.de>. Version: May 2010
- [Mar06] MARCHIONINI, Gary: Exploratory Search: From Finding to Understanding. In: *Commun. ACM* 49 (2006), Nr. 4, S. 41–46. – ISSN 0001–0782
- [Rot10] ROTH, Philipp: *Facebook Infografik - 500 Millionen Nutzer und Facebook Nutzung in Deutschland*. http://facebookmarketing.de/zahlen_fakten/infografik-500-millionen-nutzer. Version: Juli 2010
- [SH08] SIORPAES, Katharina ; HEPP, Martin: Games with a Purpose for the Semantic Web. In: *IEEE Intelligent Systems* 23 (2008), Nr. 3, S. 50–60. – ISSN 1541–1672
- [Sur04] SUROWIECKI, James: *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*. Doubleday, Anchor, 2004
- [UKOH09] URBANI, Jacopo ; KOTOULAS, Spyros ; OREN, Eyal ; HARMELEN, Frank: Scalable Distributed Reasoning Using MapReduce. In: *ISWC '09: Proceedings of the 8th International Semantic Web Conference*. Berlin, Heidelberg : Springer, 2009. – ISBN 978–3–642–04929–3, S. 634–649
- [Urb09] URBANI, Jacopo: *RDFS/OWL reasoning using the MapReduce framework*, VU University Amsterdam, Diplomarbeit, 2009
- [Wik10a] WIKIPEDIA: *DBpedia*. <http://de.wikipedia.org/wiki/DBpedia>. Version: April 2010
- [Wik10b] WIKIPEDIA: *Facebook*. <http://en.wikipedia.org/wiki/Facebook>. Version: August 2010

- [Wik10c] WIKIPEDIA: *Präattentive Wahrnehmung*. http://de.wikipedia.org/wiki/Präattentive_Wahrnehmung. Version: June 2010
- [Wik10d] WIKIPEDIA: *Die Weisheit der Vielen*. http://de.wikipedia.org/wiki/Die_Weisheit_der_Vielen. Version: July 2010
- [Wik10e] WIKIPEDIA: *Yovisto*. <http://de.wikipedia.org/wiki/Yovisto>. Version: July 2010
- [WSHK10] WAITELONIS, Jörg ; SACK, Harald ; HERCHER, Johannes ; KRAMER, Zalan: Semantically Enabled Exploratory Video Search. In: *SEMSEARCH '10: Proceedings of the 3rd International Semantic Search Workshop*. New York, NY, USA : ACM, 2010. – ISBN 978-1-4503-0130-5, S. 1-8