

Bachelorarbeit

Automatische Vorschläge zur Vervollständigung von Wikipedia Listen - Listenerkennung, Benutzeroberfläche

Robin Tibor Schirrmeister

18. April 2012



Albert-Ludwigs-Universität Freiburg im Breisgau
Technische Fakultät
Institut für Informatik

Eingereichte Bachelorarbeit gemäß den Bestimmungen der Prüfungsordnung der Albert-Ludwigs-Universität Freiburg für den Studiengang Bachelor of Science (B.Sc.) Informatik vom 31.8.2010.

Bearbeitungszeitraum

18.01.2012 – 18.04.2012

Gutachterin

Prof. Dr. Hannah Bast

Der frühe Vogel müsste den Wurm nicht gleich fangen, er
könnte ihn erstmal aufwecken :)

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Inhaltsverzeichnis

Zusammenfassung	1
1. Einleitung	3
1.1. Motivation	3
1.2. Wikipedia-Listen	3
1.3. Ablauf von WikiListSuggest	4
1.4. Listenerkennung auf Wikipedia Artikeln	4
1.5. Listenerweiterung	5
1.6. Benutzeroberfläche	7
1.7. Evaluation	8
2. Verwandte Arbeiten	9
2.1. Überblick	9
2.2. Datensätze	9
2.3. Wettbewerbe	9
2.4. TREC Entity List Completion	10
2.5. Yahoo SemSearch Challenge	10
2.6. INEX	10
2.7. Nutzen	11
3. Verwendete semantische Technologien	13
3.1. Warum Semantik verstehen?	13
3.2. YAGO	14
3.3. Broccoli	15
3.4. Part of Speech	16
3.5. Synonyme	16
4. Listenerkennung	17
4.1. Überblick	17
4.2. Erkennungsalgorithmus	18
4.3. Kandidaten-Extraktion	18
4.4. Kandidaten-Bewertung	19
4.4.1. Oberbegriffe der Kandidaten	19
4.4.2. Bewertung der Oberbegriffe	19
4.5. Listenstruktur-Erkennung	21
4.5.1. Einteilung der Kandidaten in Gruppen	22

4.5.2. Bewertung der Gruppen	22
5. Evaluation der Listenerkennung	23
5.1. Überblick	23
5.2. Methodik	23
5.3. Ergebnisse	24
5.4. Gründe für Erkennung oder Nichterkennung	25
6. Zukünftige Arbeit - Listenerkennung	27
6.1. Zukünftige Arbeit	27
6.2. Verbesserungen bei HTML-Listenelementen	27
6.3. Verbesserungen bei kleinen Listen	28
6.4. Verbesserungen bei Tabellen	28
6.5. Verbesserung beim Themenwort-Vergleich	28
6.5.1. Potential der verschiedenen Ansätze	28
7. Benutzeroberfläche	31
7.1. Funktionalitäten	31
7.2. Typischer Use Case	31
7.3. Einbindung und Start	31
7.4. Qualitätscheck	32
7.5. Anfrageverbesserung	33
7.6. Überprüfung der zusätzlichen Elemente	34
7.7. Anpassbarkeit der Oberfläche	35
8. Benutzeroberfläche - Implementierung	37
8.1. Überblick	37
8.2. Einbindung und Start	38
8.3. Qualitätscheck	40
8.4. Anfrageverbesserung	41
8.5. Überprüfen zusätzlicher Elemente	41
8.6. Zukünftige Arbeit	41
9. Zusammenfassung und Schlussfolgerungen	43
9.1. Momentaner Stand von WikiListSuggest	43
9.2. Ansatzpunkte für Verbesserungen	43
9.2.1. Listenerkennung	43
9.2.2. Benutzeroberfläche	44
9.3. Evaluation durch Nutzer	44
Danksagung	45
A. Anhang	47
A.1. Evaluationslisten	47

Zusammenfassung

Diese Arbeit beschreibt das von Simon Skilevic und mir erstellte Programm WikiListSuggest, das automatische Vorschläge zur Vervollständigung von Wikipedia-Listen generiert. In der Online-Enzyklopädie Wikipedia gibt es viele Listen zu bestimmten Themen, z.B. die *Liste essbarer Blätter*. Diese Wikipedia-Artikel werden von Hand erstellt und können unvollständig sein. Unser Programm WikiListSuggest versucht automatisch Vorschläge zur Erweiterung solcher existierenden Artikel zu erstellen. Dazu versucht es im Wikipedia-Artikel die tatsächliche Liste zu erkennen und, mit Hilfe der semantischen Suchmaschine Broccoli, Vorschläge für weitere in die Liste gehörende Elemente zu generieren. In dieser Arbeit wird hauptsächlich die automatisierte Erkennung der Liste sowie die Benutzeroberfläche beschrieben.

1. Einleitung

1.1. Motivation

Eine der großen Herausforderungen für die Informatik von heute ist es, die große Menge an bestehendem Wissen im Internet für Menschen leicht zugänglich zu machen. Information leicht zugänglich machen soll hierbei bedeuten, dass man im Internet in möglichst kurzer Zeit möglichst viele für einen selbst wichtige und interessante Informationen finden kann. Dies kann heißen, dass man eine Antwort auf eine ganz konkrete Frage sucht und diese möglichst schnell beantwortet haben will. Es kann auch heißen, dass man sich umfassend mit einem Thema auseinander setzen möchte und dabei viele für das eigene Interesse bedeutsame Dinge in Erfahrung bringen möchte.

Um die Zugänglichkeit von Informationen zu verbessern, gibt es heutzutage zum Einen automatische Ansätze wie z.B. Suchmaschinen (Bing, Google...). Zum Anderen gibt es Ansätze, die darauf aufbauen, dass Menschen selbst die Information aufbereiten, wie beispielsweise die Erstellung von Online-Enzyklopädien, einschließlich der sehr bekannten Seite Wikipedia. Beide Ansätze erlauben es den Internetnutzern, schneller und einfacher für sie relevantes Wissen im Web zu finden.

Unser Projekt WikiListSuggest befindet sich an der Schnittstelle zwischen automatischen und manuellen Techniken: Wir nehmen eine von Menschen erstellte Liste aus der englischen Wikipedia, wie z.B. die Liste essbarer Blätter, und versuchen automatisch Erweiterungsvorschläge zu machen. Diese Vorschläge können dann wiederum von einem menschlichen Nutzer in den Artikel eingepflegt werden.

Diese Listen können für einen Nutzer wertvolle Informationen gut aufbereitet enthalten, so kann eine Medizinstudentin an der Liste der virenhemmenden Medikamente interessiert sein oder jemand sprachenbegeistertes von der Liste der einheimischen Sprachen Argentiniens profitieren. So trägt die Verbesserung dieser Themensammlungen einen kleinen Teil zu einem einfacheren Zugriff auf das durch Menschen im Internet produzierte Wissen bei.

1.2. Wikipedia-Listen

Listen auf Wikipedia eignen sich aus mehreren Gründen sehr gut für die Aufgabe von Menschen erstellte Listen im Internet zu vervollständigen:

- **Praktische Relevanz:** Aufgrund der hohen Popularität Wikipedias sind die Listen in der Praxis von Bedeutung.
- **Unterschiedlichkeit:** Die Listen reichen von der Bedeutung her von relativ simpel (Liste der Muskeln im menschlichen Körper) bis sehr kompliziert (Liste der Schauspieler, die einen Oscar für eine nicht-englischsprachige Performance bekommen haben).
- **Strukturelle Ähnlichkeit:** Die meisten Listen auf Wikipedia haben eine ähnliche Struktur, was ihre Erkennung erleichtert.
- **Gute Datenbasis für Suche:** Der Inhalt von Wikipedia bietet auf Grund der im Vergleich zum restlichen Web hohen Qualität der Texte eine gute Möglichkeit, weitere Elemente zu finden.

1.3. Ablauf von WikiListSuggest

Als erstes muss WikiListSuggest die tatsächlichen *Listenelemente* in dem Wikipedia-Listen-Artikel erkennen. Danach probiert die Software, eine Anfrage an die semantische Suchmaschine *Broccoli* zu erstellen. Diese Anfrage soll möglichst die vollständige Liste zu dem gesuchten Thema finden, also im Idealfall die bisherigen Elemente abdecken und weitere passende Elemente finden. Dieser Teil wird in [Ski12] näher erläutert. Zum Schluss müssen die Ergebnisse dem Nutzer verständlich und übersichtlich dargestellt werden. Die hierfür gestaltete Benutzeroberfläche wird in Kapitel 7 und Kapitel 8 genauer beschrieben.

1.4. Listenerkennung auf Wikipedia Artikeln

List of civil rights leaders

From Wikipedia, the free encyclopedia

Below is a list of some (chiefly [American](#)) [civil rights](#) leaders:

- [Abernathy, Ralph](#) (1926–1990) clergyman, activist, [Southern Christian Leadership Conference](#) (SCLC) official
- [Anthony, Susan B.](#) (1820–1906) women's suffrage/voting rights leader
- [Baker, Ella](#) (1903–1986) Member of Student Nonviolent Coordinating Committee (SNCC)
- [Bates, Daisy](#) (1914–1999)
- [Bevel, James](#) (1936–2008) SCLC's main strategist, organizer, and Direct Action leader

Abbildung 1.1.: Ausschnitt aus der Liste der Bürgerrechtler

Die Erkennung der Listenelemente läuft meist auf das Finden einer Struktur im Wikipedia-Artikel heraus, in der sich alle Listenelemente befinden. Betrachten wir hierzu einige Beispiele, zunächst in Abbildung 1.1 eine von der Struktur her einfache Liste, eine Liste von Bürgerrechtlern. Die Listenelemente befinden sich hier immer als erste Links in einem HTML-Listen-Element.

Listen auf Wikipedia können jedoch auch verschachtelt sein, wie bei der *Liste essbarer Samen* in Abbildung 1.2.

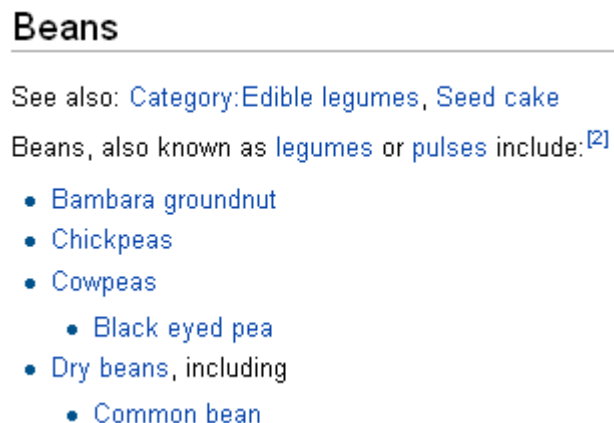


Abbildung 1.2.: Ausschnitt aus der Liste der essbaren Samen

Die Listenelemente können sich auch in anderen Strukturen, wie z.B. Tabellen befinden. Dies sieht man z.B. an der *Liste der größten Wüsten* in Abbildung 1.3. Für diese Liste ist es nicht nur notwendig zu erkennen, dass die Listenelemente in der Tabelle enthalten sind, sondern auch, dass sie in der zweiten Spalte sind und nicht in der letzten, wo die Orte der Wüsten stehen.

Insgesamt befindet sich die tatsächliche Liste also nicht bei allen Wikipedia-Listen-Artikeln an der gleichen Stelle. Es lässt sich allerdings meistens eine Struktur innerhalb eines Wikipedia-Listen-Artikels finden, in dem sich die komplette Liste befindet. Wie diese definiert und gefunden werden kann, wird in Kapitel 4 genauer erläutert.

1.5. Listenerweiterung

Nachdem die Liste auf dem Wikipedia-Artikel erkannt wurde, versucht WikiListSuggest Vorschläge für die Erweiterung der Liste machen. Hierfür versucht es, die vollständige zum Thema passende Liste zu finden. So würde WikiListSuggest bei der Liste der Bürgerrechtler versuchen, alle in Wikipedia vorhandenen Bürgerrechtler zu finden, inklusive solcher, die in der Wikipedia-Liste fehlen.

Um diese vollständige Liste zu finden, benutzt WikiListSuggest die Suchmaschine *Broccoli*. Broccoli ermöglicht es einerseits wie bei einer "klassischen" Volltext-Suchmaschine

List of deserts by area

From Wikipedia, the free encyclopedia

This is a **list of deserts in the world by area**. It includes all deserts with an area greater than 50,000 square kilometres (19,300 sq mi).

The List

[\[edit\]](#)

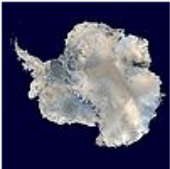
Rank ↕	Name ↕	Type ↕	Image ↕	Area (km²) ↕	Area (sq mi) ↕	Location ↕
1	Antarctic Desert	Polar		13,829,430 ^[1]	5,339,573	Antarctica
2	Arctic	Polar		13,726,937 ^[2]	5,300,000	Alaska (United States) , Canada , Finland , Greenland (Denmark) , Iceland , Norway , Russia and Sweden
3	Sahara	Subtropical		9,100,000+ ^[3]	3,320,000+	Algeria , Chad , Egypt , Eritrea , Libya , Mali , Mauritania , Morocco , Niger , Sudan , Tunisia and Western Sahara

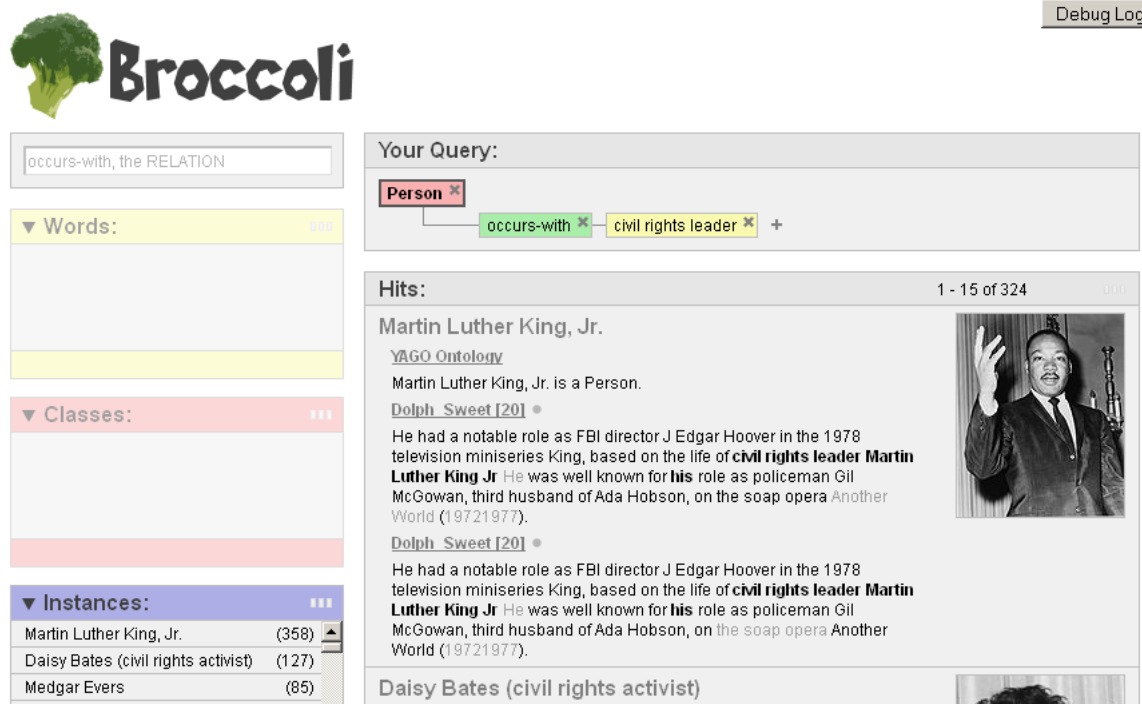
Abbildung 1.3.: Ausschnitt aus der Liste der größten Wüsten

nach bestimmten Suchwörtern zu suchen. Andererseits erlaubt es zusätzlich noch zu bestimmen, zu welchem *Oberbegriff* die Suchergebnisse gehören sollen. Eine Suchanfrage für Bürgerrechtler könnte man z.B. mit dem Oberbegriff *Person* und den Suchwörtern *civil rights leader* stellen (*Bürgerrechtler* selbst kennt Broccoli nicht als Oberbegriff).

Diese Anfrage sieht man in der Benutzeroberfläche von Broccoli in Abbildung 1.4. In der Mitte oben ist die Anfrage: In Rot der Oberbegriff *Person* und durch das grüne *occurs-with*-Kästchen verbunden die Suchwörter *civil rights leader*. Darunter befinden sich die Suchergebnisse samt Textauszügen, in denen sie in Wikipedia mit den Wörtern *civil rights leader* vorkommen.

Man kann die Suchwörter auch benutzen, wenn man nur Suchergebnisse mit bestimmten Eigenschaften sucht: So kann man z.B. nach Pflanzen mit essbaren Blättern durch den Oberbegriff *Pflanze* und den Suchwörtern *essbar* und *Blätter* suchen.

WikiListSuggest versucht nun, eine Anfrage an Broccoli zu erstellen, die möglichst gut die vollständige Liste zum gesuchten Thema finden soll. Die Elemente des Broccoli-Ergebnisses, die in der Wikipedia-Liste fehlen, werden dann als Vorschläge



The screenshot shows the Broccoli web application interface. At the top left is the Broccoli logo (a green broccoli head). At the top right is a 'Debug Log' button. Below the logo is a search bar containing the text 'occurs-with, the RELATION'. To the left of the main content area are three panels: 'Words:' (empty), 'Classes:' (empty), and 'Instances:' (listing 'Martin Luther King, Jr.' (358), 'Daisy Bates (civil rights activist)' (127), and 'Medgar Evers' (85)). The main content area is titled 'Your Query:' and shows a query graph with 'Person' as the root, connected to 'occurs-with', which is connected to 'civil rights leader'. Below this is a 'Hits:' section showing search results. The first result is 'Martin Luther King, Jr.' with a photo and a description: 'He had a notable role as FBI director J Edgar Hoover in the 1978 television miniseries King, based on the life of civil rights leader Martin Luther King Jr. He was well known for his role as policeman Gil McGowan, third husband of Ada Hobson, on the soap opera Another World (1972-1977)'. The second result is 'Daisy Bates (civil rights activist)' with a partial photo.

Abbildung 1.4.: Broccoli-Anfrage für Bürgerrechtler

für die Erweiterung angezeigt.

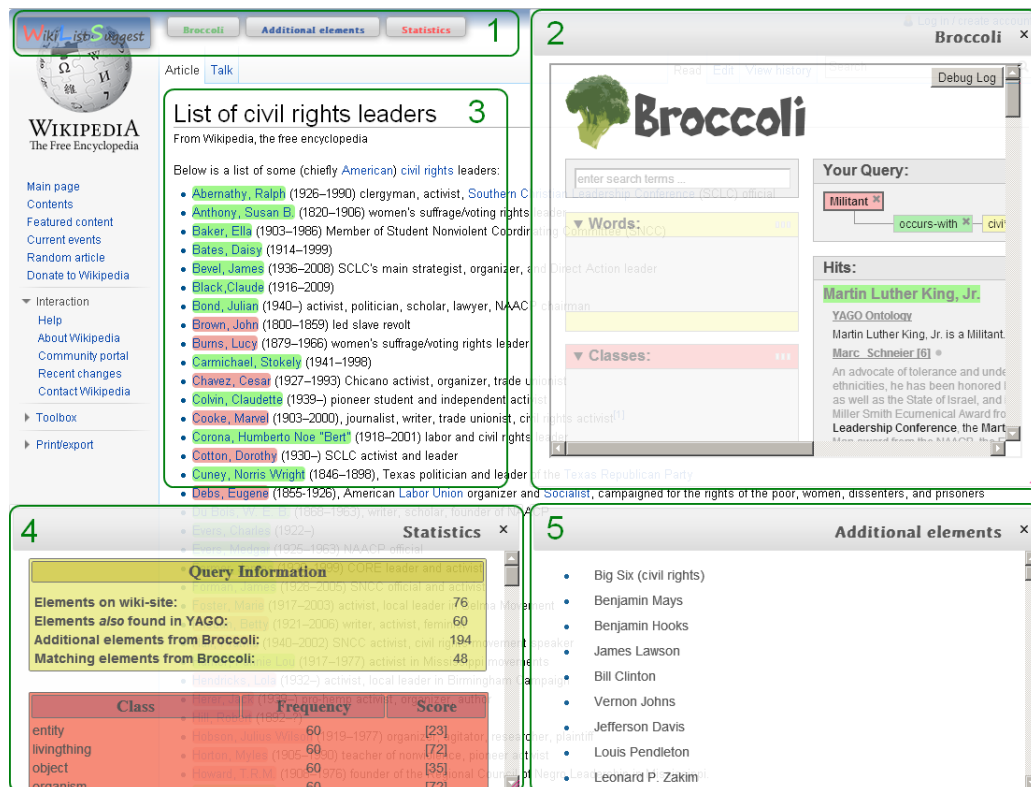
1.6. Benutzeroberfläche

Zum Schluss müssen die Ergebnisse dem Nutzer angezeigt werden. Die Benutzeroberfläche von WikiListSuggest soll hierbei folgende Möglichkeiten bieten:

- Die Erweiterungsvorschläge sehen
- Eine Begründung sehen, wieso die Erweiterungsvorschläge in die Liste passen könnten
- Die Anfrage an Broccoli manuell verändern können
- Die im Wikipedia-Artikel erkannte Liste sehen, um Fehler bei der Erkennung ausschließen zu können
- Die von der erkannten Wikipedia-Liste durch die Anfrage wiedergefundenen und nicht wiedergefundenen Listenelemente zu sehen, um die Qualität der Anfrage abschätzen zu können

Wir haben die Benutzeroberfläche hierfür so umgesetzt, dass sie sich in eine Wikipedia-Seite einbindet: Beim Surfen auf einer Wikipedia-Listen-Seite kann unser Programm aufgerufen werden und die Benutzeroberfläche mit den genannten Funktionalitäten

erscheint. In Abbildung 1.5 ist die Oberfläche auf der Liste der Bürgerrechtler zu sehen.



- 1 WikiListSuggestMenü mit Buttons zum Öffnen der Fenster: Broccoli (Broccoli-Oberfläche), Additional Elements (zusätzliche von Broccoli gefundene Elemente) und Statistics (Statistiken über die Listenelemente und die Broccoli-Anfrage)
- 2 Broccoli-Oberfläche mit vorgeschlagener Anfrage
- 3 Erkannte Listenelemente, grün, wenn auch von Broccoli-Anfrage gefunden, rot, wenn nicht
- 4 Statistiken über die Wikipedia-Liste und die Broccoli-Anfrage
- 5 Zusätzliche, von Broccoli gefundene Elemente

Abbildung 1.5.: Screenshot der WikiListSuggest-Benutzeroberfläche

1.7. Evaluation

Eine Evaluation der Listenerkennung wird in Kapitel 4 beschrieben. Eine ausführlichere Evaluation von WikiListSuggest insgesamt gibt es in [Ski12].

2. Verwandte Arbeiten

2.1. Überblick

Die Aufgabe der *Listenvervollständigung*, also zu einer gegebenen Menge von Listenelementen weitere passende Elemente zu finden, wird natürlich nicht nur von uns bearbeitet. Hierbei muss man sich auch nicht wie wir auf Daten aus Wikipedia oder auf einen bestimmten Suchansatz (wie Oberbegriffe und Volltextsuche) beschränken.

Im Folgenden folgt eine kurze Vorstellung einiger alternativer Datensätze und einiger Wettbewerbe, bei denen Gruppen versuchen, mit Hilfe dieser Datensätze möglichst gut Listen zu vervollständigen.

2.2. Datensätze

Einige für die Listenvervollständigung benutzte Datensätze sind:

- Daten aus Wikipedia(so wie unser Projekt)
- Den Text von Webseiten des gesamten World Wide Web
- Auf Webseiten hinterlegte Meta-Daten: Mit verschiedenen maschinenlesbaren Datenformaten können im Web prinzipiell über alle möglichen Dinge Daten hinterlegt werden, z.B. wer eine bestimmte Seite geschrieben hat, was die Hauptstadt von Kamerun ist, etc.

Natürlich können alle diese Datensätze auch beliebig miteinander kombiniert werden.

2.3. Wettbewerbe

Für die Aufgabe der Listenvervollständigung gibt es mehrere Wettbewerbe, die solche Datensätze benutzen, z.B.:

- *TREC Entity List Completion*: Hier wird sowohl auf Websites allgemein gesucht als auch auf bereits aufbereiteten, im Web gefundenen Meta-Daten
- *Yahoo SemSearch Challenge*: Hier wird nur eine extra für solche Wettbewerbe ausgesuchte, möglichst allgemeine Menge von im Web gefunden Metadaten benutzt

- *INEX*: Hier wird nur auf Wikipedia-Daten gesucht

2.4. TREC Entity List Completion

Beim letztmals 2011 ausgeführten TREC Entity List Completion Wettbewerb¹ sah die Eingabe so aus:

- Input-Entität, definiert durch Name und Homepage, z.B. *Organization of Petroleum Exporting Countries (OPEC)*, <http://www.opec.com/>
- Typ der Listenelemente, z.B. Länder
- Beschreibung der Liste, z.B. Länder, die Mitglied von OPEC sind
- Beispiele aus der Liste, z.B. Katar

Auf Basis dieser Eingabe sollten dann weitere Listenelemente gefunden werden. Dafür durfte man sowohl in bereits vorher im Web gefundenen und aufbereiteten Metadaten suchen als auch auf einer 25 Terabyte großen Sammlung heruntergeladener Webseiten. Leider nahmen 2011 nur wenige Teams an diesem Wettbewerb teil, so dass er dieses Jahr nicht stattfinden wird [BSV11].

2.5. Yahoo SemSearch Challenge

Der Listenvervollständigungsteil der Yahoo SemSearch Challenge² wurde nur auf im Web gefundenen Metadaten durchgeführt. Hier war die Eingabe nur eine Freitext-Suchanfrage wie *Arabische Staaten am Persischen Golf*. Auf Basis dieser Suchanfrage sollte man dann weitere zur Suchanfrage passende Entitäten finden.

2.6. INEX

Bis zum Jahr 2009 gab es auch noch den INEX Entity List Completion Wettbewerb³. Bei diesem war die Eingabe:

- Titel der gesuchten Liste: z.B. *Europäische Länder mit Euros*
- Beschreibung: z.B. *Ich will eine Liste mit Europäischen Ländern wo ich mit Euros bezahlen kann*
- Kontext: z.B. *Jeder Antwort sollte ein Artikel über ein bestimmtes Europäisches Land sein, in dem man mit Euros bezahlen kann*
- Beispiele: *Frankreich, Deutschland, Spanien*

¹<http://ilps.science.uva.nl/trec-entity/>

²<http://semsearch.yahoo.com/>

³<http://www.inex.otago.ac.nz/tracks/entity-ranking/entity-ranking.asp>

2.7. Nutzen

Der Nutzen dieser und ähnlicher Wettbewerbe ist nicht nur die Möglichkeit, bessere Ansätze für Listenvervollständigung zu finden. Neben diesem Nutzen für das spezielle Listenvervollständigungsproblem erlauben diese Wettbewerbe auch allgemeine Abschätzungen über die Qualität bestimmter semantischer Suchansätze oder semantischer Suchmaschinen und können Hinweise darauf geben, was besser und was schlechter funktioniert.

So kann auch WikiListSuggest und dessen Evaluation Hinweise zum Weiterentwickeln und Evaluieren von Broccoli geben.

3. Verwendete semantische Technologien

3.1. Warum Semantik verstehen?

Um Listen erkennen und erweitern zu können, braucht es oft ein Verständnis der Semantik (Bedeutung) von bestimmten Wörtern. Man betrachte hierzu wieder einen Ausschnitt aus der Wikipedia-Liste der größten Wüsten:

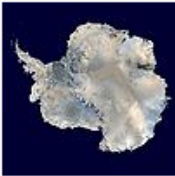
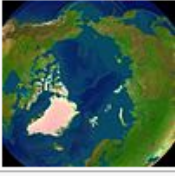
1	Antarctic Desert	Polar		13,829,430 ^[1]	5,339,573	Antarctica
2	Arctic	Polar		13,726,937 ^[2]	5,300,000	Alaska (United States) , Canada , Finland , Greenland (Denmark) , Iceland , Norway , Russia and Sweden
3	Sahara	Subtropical		9,100,000+ ^[3]	3,320,000+	Algeria , Chad , Egypt , Eritrea , Libya , Mali , Mauritania , Morocco , Niger , Sudan , Tunisia and Western Sahara

Abbildung 3.1.: Ausschnitt aus der Liste der größten Wüsten

Als Menschen wissen wir nun wahrscheinlich, dass in der zweiten Spalte die Wüsten stehen. Wir wissen dies, da wir die Bedeutung der Wörter kennen: Entweder kennen wir einen der Namen der Wüsten oder wir wissen, dass in der letzten Spalte größtenteils Staaten stehen, also keine Wüsten. Um die Erkennung zu automatisieren, brauchen wir also Programme, die Wörter bestimmten Oberbegriffen zuordnen können (*Sahara* → *Wüste*, *Kanada* → *Staat*, ...). Auch um automatisch Vorschläge für die Erweiterung der Liste machen zu können, muss unser Programm wissen, ob eine gefundene große Region eine Wüste ist oder nicht.

Für die Erweiterung der Liste kann auch ein Verständnis vom Aufbau von natürlicher Sprache notwendig sein. Betrachten wir zum Beispiel den Text:

Die Sahara ist eine Wüste in Afrika. Ihre große Ausdehnung macht *sie* zu einer der größten Wüsten der Welt.

Hier kann nur erkannt werden, dass die Sahara eine große Wüste ist, wenn vorher bestimmt wurde, das mit *sie* die Sahara gemeint ist. Dafür bedarf es Kenntnisse über natürliche Sprache.

Im Folgenden wird zunächst die Wissensdatenbank *YAGO* (Yet another Great Ontology) vorgestellt. Mit ihr ist unter anderem die Zuordnung von Wörtern zu Oberbegriffen möglich. Danach beschreiben wir weitere Details der semantischen Suchmaschine Broccoli, die das Wissen von *YAGO* und Kenntnisse über natürliche Sprache mit "klassischer" Volltextsuche kombiniert.

3.2. YAGO

Wie erklärt, brauchen wir an verschiedenen Stellen von WikiListSuggest eine Möglichkeit, automatisch Wörter bestimmten Oberbegriffen zuzuordnen. Eine gute Technologie dafür sind *Ontologien* wie *YAGO*¹: Wissensdatenbanken, in denen Beziehungen zwischen Wörtern (bzw. diesen Wörtern zugeordneten Entitäten) gespeichert sind. Für Oberbegriffe wäre dann z.B. in solch einer Ontologie gespeichert:

- **Sahara** *ist eine Wüste*
- **Deutschland** *ist ein Staat*
- **Apfel** *ist eine Frucht*
- **Michael Jackson** *ist ein Sänger*

In Ontologien können jedoch auch weitere Beziehungen gespeichert sein wie:

- **Berlin** *befindet sich in Deutschland*
- **Penicillin** *wurde entdeckt von Alexander Fleming*
- **Martin Luther King** *wurde geboren 1929*

YAGO ist eine solche Ontologie, die mehr als 10 Millionen Entitäten und mehr als 120 Millionen Beziehungen zwischen diesen Entitäten kennt.

Die Beziehungen zwischen den Entitäten könnten bei der Suche nach neuen Listenelementen nützlich sein, wenn man z.B. eine Liste von Schauspielern aus Berlin sucht, so kann man nur Personen, die in Berlin geboren sind, in die Ergebnisse aufnehmen.

¹<http://www.mpi-inf.mpg.de/yago-naga/yago/>

3.3. Broccoli

Broccoli ist eine Suchmaschine, die das Wissen aus einer Ontologie wie YAGO mit der Volltextsuche nach beliebigen Wörtern vereint. So erlaubt sie dem Nutzer, mehrere Suchmöglichkeiten zu kombinieren:

- Suche nach Oberbegriffen, z.B. die Suche nach allen Pflanzen oder nach allen Schauspielern
- Einschränkung des Suchergebnisses durch bestimmte, der Wissensdatenbank bekannte, Eigenschaften der Begriffe: nur Pflanzen aus Deutschland, nur Schauspieler, die einen Oscar gewonnen haben, ...
- Einschränkung des Suchergebnisses durch beliebige Wörter, die in Wikipedia mit dem Begriff zusammen auftauchen: Nur Pflanzen, die mit *essbare Blätter* auftauchen, nur Schauspieler, die mit *erfolgreich* auftauchen,...

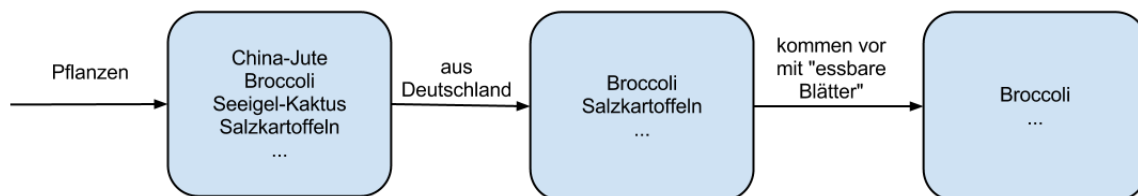


Abbildung 3.2.: Mögliche Anfrage an Broccoli: Zunächst werden alle Entitäten eines bestimmten Oberbegriffes (*Pflanze*) genommen. Dieses Ergebnis kann eingeschränkt werden durch: Nur Entitäten mit bestimmten Eigenschaften (aus Deutschland) oder nur Entitäten, die mit bestimmten Suchwörtern vorkommen (*essbare Blätter*).

Die Suchwörter sind insbesondere deswegen hilfreich, weil jede Ontologie immer nur eine begrenzte Anzahl an Beziehungen zwischen Entitäten enthalten kann. Nehmen wir an, jemand will wissen, welche Pflanzen aus Deutschland essbare Blätter haben. Eine Ontologie hat wahrscheinlich die Information gespeichert, welche Pflanzen es gibt und könnte als Beziehung auch wissen, wo welche Pflanze heimisch ist. Nur mit Hilfe einer Ontologie könnte man also bereits die in Deutschland heimischen Pflanzen finden.

Eine Ontologie weiß allerdings höchstwahrscheinlich nichts darüber, welche Pflanzen essbare Blätter haben oder nicht. Eine Idee, Pflanzen mit essbaren Blättern zu finden, ist nun, das Wissen der Ontologie mit einer Suche nach den Wörtern *essbar* und *Blätter* zu kombinieren.

Broccoli ermöglicht dies: Wenn es in einem Text ein Wort als Pflanze aus Deutschland identifiziert, kann es im gleichen Kontext nach den Wörtern *essbar* und *Blätter* suchen. Ein Verständnis natürlicher Sprache und das Beachten des gleichen Kontextes spielen beide hierbei eine große Rolle: Der Satz

Der Strunk des Rhabarbers ist essbar, aber seine Blätter sind giftig.

illustriert beides auf einmal: Erstens muss Broccoli erkennen können, dass *seine* sich auf Rhabarber bezieht. Zweitens muss Broccoli merken, dass Blätter und essbar im gleichen Satz, aber in verschiedenen *Kontexten* auftreten. Durch den zweiten Schritt wird verhindert, dass Broccoli Rhabarber als Pflanze mit essbaren Blättern identifiziert. Genauer wird Broccoli in [BBBH12] vorgestellt.

3.4. Part of Speech

Um zu entschlüsseln, zu welchem Oberbegriff die Listenelemente einer Liste gehören, kann ebenso ein Verständnis von Sprache hilfreich sein. Wenn man die Listentitel

- List of *deserts* by area
- List of leaf *vegetables* from Germany
- List of highest-grossing *movies*

betrachtet, fällt auf, dass die Oberbegriffe der Listenelemente immer das erste Nomen im Plural sind. Um solche Sprachbestandteile zu identifizieren, gibt es so genannte *Part of Speech-Tagger*. Wir benutzen bei unserer Arbeit den Illinois Part of Speech-Tagger².

3.5. Synonyme

Auch Synonyme können nützlich sein, z.B. sollte eine Entität sowohl als Film identifiziert werden, wenn sie in der verwendeten Ontologie unter *movie* vermerkt ist, als auch wenn sie unter *film* vermerkt ist. Für die Synonyme verwenden wir die Wordnet-Datenbank³.

²http://cogcomp.cs.illinois.edu/page/software_view/POS

³<http://wordnet.princeton.edu/>

4. Listenerkennung

4.1. Überblick

Wir betrachten bei der Listenerkennung der Einfachkeit halber nur solche Listenelemente, die einen Wikipedia-Artikel haben. Um die Listenelemente finden zu können, betrachtet unser Programm einerseits, welche Wörter zu welchen Oberbegriffen gehören. Andererseits muss es auch die Struktur des Wikipedia-Listen-Artikels beachten.

Bei der Liste der Wüsten würde es beispielsweise prinzipiell ausreichen, nur die Oberbegriffe zu beachten: Alle auf dem Wikipedia-Listen-Artikel vorkommenden Wüsten sind Listenelemente. Bei vielen Listen reicht die Betrachtung der Oberbegriffe jedoch nicht aus, da kein passender Oberbegriff genau die Listenelemente beschreibt. Zur Veranschaulichung ein Ausschnitt aus dem Listen-Artikel namhafter tauber Menschen:

The success of the Paris school spawned similar schools throughout [Europe](#) and the [New World](#). Significant in [American history](#), deaf Frenchman [Laurent Clerc](#), both a student and teacher at the Paris school (1798–1816), and Rev. [Thomas Hopkins Gallaudet](#), a hearing American, founded the first school for the deaf in [Hartford, Connecticut](#) in 1816. Clerc, along with Paris school faculty members [Jean Massieu](#) and [Ferdinand Berthier](#) formed the core of a group of pioneering deaf intellectuals. They are joined by many people on this list who, like them, were born deaf, used a [sign language](#) as their mother [language](#), and were notable for their leadership and accomplishments.

- [Ferdinand Berthier](#), French intellectual, published several articles, first deaf person to receive the French Legion of Honor, founder of world's first deaf organization
- [Julia Brace](#) (1807–1884), early American [deaf-blind](#) student at the [Hartford School for the Deaf](#)
- [John Brewster Jr.](#) (1766–1854), American, itinerant artist of the Federalist Period in America
- [Laura Bridgman](#), (1829–1889), American, first deaf-blind student of Dr. Samuel Howe at the [Perkins School for the Blind](#)
- [Laurent Clerc](#) (1785–1869), French-American, co-founder of first school for the deaf in America, first deaf teacher of the deaf in America

Abbildung 4.1.: Ausschnitt aus der Liste namhafter tauber Menschen

Nur mit Hilfe der Oberbegriffe könnte man als ersten Versuch hier alle Menschen als Listenelemente nehmen. Dann würde man allerdings falsche Treffer aus dem über der echten Liste stehenden Text erhalten, in dem auch nicht-taube Menschen vorkommen, z.B. *Thomas Hopkins Gallaudet*. Leider gibt es keinen YAGO bekannten Oberbegriff, unter dem tatsächlich alle tauben Menschen eingespeichert sind. Ein Weg, die Listenelemente hier zu erkennen, ist die HTML-Struktur des Artikels und das Wissen

aus YAGO zu kombinieren: Zunächst wird die *Teilstruktur* des Artikels identifiziert, in der sich mit höchster Wahrscheinlichkeit die Listenelemente befinden. Hier wäre diese Teilstruktur z.B. die jeweils ersten Links in der Aufzählungspunkte-Liste unter dem Text, da sich in ihnen die meisten Links auf Menschen befinden. Dann werden alle Wikipedia-Links innerhalb dieser Teilstruktur als Listenelemente betrachtet.

4.2. Erkennungsalgorithmus

Insgesamt ist der Ablauf der Listenerkennung bei WikiListSuggest in 3 Schritte unterteilt:

1. **Kandidaten-Extraktion:** Extrahiere aus dem Wikipedia-Artikel alle *Kandidaten*, also alle Wikipedia-Links, die Listenelemente sein könnten
2. **Kandidaten-Bewertung:** Berechne für alle diese Kandidaten die Wahrscheinlichkeit, dass sie Teil der Liste sind
3. **Listenstruktur-Erkennung:** Identifiziere mit Hilfe dieser Wahrscheinlichkeiten die HTML-Struktur, in der sich wahrscheinlich alle Listenelemente befinden und gebe alle Kandidaten innerhalb dieser Struktur als gefundene Liste zurück

4.3. Kandidaten-Extraktion

Der erste Schritt der Erkennung der Listenelemente ist also, die möglichen Kandidaten auf der Seite zu bestimmen. Wir nehmen dabei als Kandidaten die Links auf Wikipedia-Artikel, schränken uns jedoch noch weiter ein, um keine unnötigen Kandidaten zu betrachten:

- Wir betrachten nur Links innerhalb von HTML-Listen und HTML-Tabellen. Fast alle Wikipedia-Listen sind innerhalb dieser HTML-Strukturen zu finden.
- Bestimmte Teile der Seite ignorieren wir, z.B. der Referenzen-Teil der Wikipedia-Seite.

Für alle diese Schritte benutzen wir CSS3-Selektoren¹. Zunächst selektieren wir durch die Klasse `.mw-content-ltr` den Inhaltsbereich des Wikipedia-Artikels. Auf diesem werden dann mit

```
td a[href^="/wiki/"], ul a[href^="/wiki/"],  
ol a[href^="/wiki/"], dl a[href^="/wiki/"]
```

die Kandidaten selektiert und mit Filtern wie

¹<http://www.w3.org/TR/2009/PR-css3-selectors-20091215/#selectors>, für ein leichter zugängliches Tutorial siehe <http://css.maxdesign.com.au/selectutorial/>

`h2:has(span#References) ~ * * a`

unnötige Links aus den Kandidaten herausgelöscht (hier die Links nach der Referenzen-Überschrift).

4.4. Kandidaten-Bewertung

Wenn WikiListSuggest eine Menge von Kandidaten erfasst hat, muss es als nächstes die Wahrscheinlichkeit berechnen, dass ein bestimmter Kandidat tatsächlich zur Liste gehört. Dazu wird versucht herauszufinden, durch welche Oberbegriffe die Listenelemente gut beschrieben werden:

1. Alle Oberbegriffe, zu denen die Kandidaten für die Listenelemente gehören, werden durch YAGO gefunden
2. Es wird versucht, für jeden gefundenen Oberbegriff einen *Schätzwert* zu berechnen, wie gut dieser Oberbegriff die tatsächlichen Listenelemente erfasst: So würde die Elemente der Liste von Wüsten am Besten durch *Wüste* erfasst werden, weniger gut, weil zu allgemein durch *Region*, weniger gut, weil zu speziell durch *Sandwüste* und überhaupt nicht durch *Staat*.
3. Ein Kandidat selbst wird dann umso besser bewertet, je besser die Schätzwerte seiner Oberbegriffe sind

4.4.1. Oberbegriffe der Kandidaten

Um die Oberbegriffe eines Kandidaten zu finden, werden in der Ontologie YAGO die Oberbegriffe des Kandidaten erfragt. Hierfür wird aus dem Link des Kandidaten der Name des Wikipedia-Artikels, auf den er zeigt, extrahiert. Aus dem Link `en.wikipedia.org/wiki/Albert_Einstein` wird so *Albert Einstein* extrahiert. Die Oberbegriffe von Albert Einstein (z.B. Person, Wissenschaftler, ...) sind dann die Oberbegriffe für diesen Kandidaten.

4.4.2. Bewertung der Oberbegriffe

Als nächstes werden die Oberbegriffe selbst bewertet: Ein Oberbegriff ist umso besser, je mehr tatsächliche Listenelemente zu ihm gehören und je weniger der restlichen Kandidaten zu ihm gehören. Um dies abschätzen zu können, werden vier Punkte betrachtet:

- Wie oft ein Oberbegriff als Oberbegriff eines Kandidaten vorkommt
- Wie oft ein Oberbegriff in YAGO insgesamt vorkommt

- Ob ein Oberbegriff das *Themenwort* des Titels ist: Bei der Liste namhafter tauber Menschen wäre z.B. *Menschen* das Themenwort. Wir nehmen hierbei mit Hilfe des Illinois POS-Taggers immer das erste Nomen im Plural als Themenwort.
- Ob ein Oberbegriff im Rest des Titels als Wort enthalten ist

Betrachten wir als Beispiel wieder die Liste der Wüsten in Abbildung 4.2.

List of deserts by area

From Wikipedia, the free encyclopedia

This is a **list of deserts in the world by area**. It includes all deserts with an area greater than 50,000 square kilometres (19,300 sq mi).

The List

[\[edit\]](#)

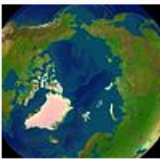
Rank ↕	Name ↕	Type ↕	Image ↕	Area (km²) ↕	Area (sq mi) ↕	Location ↕
1	Antarctic Desert	Polar		13,829,430 ^[1]	5,339,573	Antarctica
2	Arctic	Polar		13,726,937 ^[2]	5,300,000	Alaska (United States) , Canada , Finland , Greenland (Denmark) , Iceland , Norway , Russia and Sweden
3	Sahara	Subtropical		9,100,000+ ^[3]	3,320,000+	Algeria , Chad , Egypt , Eritrea , Libya , Mali , Mauritania , Morocco , Niger , Sudan , Tunisia and Western Sahara

Abbildung 4.2.: Ausschnitt aus der Liste der größten Wüsten

Bei dieser Liste führen die bei den Kandidaten häufig vorkommenden Oberbegriffe nicht direkt zu den passendsten Oberbegriffen: Sowohl die Links in der zweiten Spalte (*Antarctic Desert*, *Arctic*,...) als auch die Links in der letzten Spalte (*Antarctica*, *Alaska*,...) sind Kandidaten für die Listenelemente. Sehr viele dieser Kandidaten aus beiden Spalten haben *Ort* und *Region* als Oberbegriffe. *Wüste* hingegen haben nur Kandidaten aus der zweiten Spalte als Oberbegriff.

In diesem Fall hilft bei der Erkennung passender Oberbegriffe, dass *Region* und *Ort* zwar häufig bei den Kandidaten vorkommen, aber auch sehr häufig als Oberbegriff in YAGO selbst vorkommen: Wichtig für die Bewertung des Oberbegriffs ist also die Häufigkeit als Oberbegriff von den Kandidatenelementen relativ zur Häufigkeit in YAGO.

Auch der Titel kann für die Bewertung der Oberbegriffe nützlich sein: Hier ist z.B. der Oberbegriff *Wüste* bereits in der Mehrzahl im Titel enthalten und wird als Themenwort erkannt. So können wir unseren Schätzwert für *Wüste* als passenden Oberbegriff stark erhöhen. Für den Fall, dass das Themenwort der Liste nicht erkannt wird, betrachten wir auch noch die restlichen Wörter des Titels.

Insgesamt berechnen wir die Bewertung $s(O)$, wie passend ein Oberbegriff O die Listenelemente beschreibt, folgendermaßen:

$$s(O) = H_r(O) \cdot w(O) \quad (4.1)$$

$H_r(O)$ ist hierbei die relative Häufigkeit des Oberbegriffes, also die Häufigkeit des Oberbegriffs in der Kandidatenmenge geteilt durch die Häufigkeit des Oberbegriffes in der Ontologie. $w(O)$ ist das Gewicht des Oberbegriffes (je höher, desto eher ist O ein passender Oberbegriff für die Listenelemente).

$w(O)$ berechnet sich aus den Titelwörtern sowie einem *Ausschlussfaktor* für zu seltene Oberbegriffe der Kandidatenmenge:

$$w(O) = 1000^{x_1} \cdot 10^{x_2} \cdot 0.1^{x_3} \quad (4.2)$$

x_1 , x_2 und x_3 sind hierbei alle entweder 0 oder 1. x_1 ist 1, wenn der Oberbegriff O das Themenwort des Listentitels ist. x_2 ist 1, wenn der Oberbegriff O in den restlichen Wörtern des Titels vorkommt. x_3 ist der Ausschlussfaktor: x_3 ist 1, wenn der Oberbegriff zehnmal seltener in der Kandidatenmenge vorkommt als der häufigste Oberbegriff.

Wir multiplizieren die Faktoren hier miteinander, da wir nicht nur an der Ordnung der Kandidaten (welcher Kandidat ist wahrscheinlicher), sondern auch an der Differenz ihrer Bewertungen interessiert sind (wie viel wahrscheinlicher ist ein Kandidat als ein anderer). Die Differenz ist wichtig, da wir später Gruppen von Kandidaten betrachten: Eine Gruppe mit einem Kandidaten, der sehr wahrscheinlich ein Listenelement ist, z.B. weil einer seiner Oberbegriffe das Themenwort ist, sollte höher gewertet werden als eine Gruppe mit vielen Kandidaten, die wahrscheinlich nicht Teil der Liste sind. Dies sollte auch gelten, wenn diese Kandidaten alle nicht durch den Ausschlussfaktor ausgeschlossen werden. Gleichung 4.2 ist die einfachste von uns gefundene Funktion, die dies alles realisiert.

4.5. Listenstruktur-Erkennung

Als nächstes wird versucht, die Teilstruktur des Artikels zu ermitteln, in der sich die Listenelemente befinden. Hier können wir ausnutzen, dass die meisten Listenelemente auf Wikipedia sich in einer definierbaren Struktur befinden, bei der Liste der Wüsten in Abbildung 4.2 ist dies z.B. die 2. Spalte in der Tabelle, bei der Listennamhafter tauber Menschen die HTML-Liste mit Aufzählungspunkten.

Wir müssen die Listenelemente enthaltende Struktur auch deswegen finden, weil die Bewertung der Kandidaten nicht perfekt ist: Manche Kandidaten werden fälschlicherweise als nicht zur Liste zugehörig bewertet, weil sie z.B. gar nicht oder nicht mit allen passenden Oberbegriffen in der Ontologie vorhanden sind. Andere Kandidaten werden fälschlicherweise als zugehörig bewertet, weil sie zum gleichen Oberbegriff wie die Liste gehören. So wird bei der Liste der Wüsten sowohl die Antarktische Wüste als auch die Arktis als nicht zur Liste zugehörig bewertet, da sie in YAGO nicht als Wüsten vermerkt sind. Da die Bewertung im Durchschnitt jedoch oft richtig liegt, werden sich in der richtigen Struktur mehr hoch bewertete Kandidaten finden als in einer falschen Struktur.

Insgesamt gibt es bei der Vervollständigung zwei Schritte: Einteilung der Kandidaten in *Strukturgruppen* und Bewertung dieser Gruppen.

4.5.1. Einteilung der Kandidaten in Gruppen

Um die Kandidaten anhand der Struktur des Artikels in Strukturgruppen einzuteilen, betrachten wir nur die sie jeweils direkt umgebende Teilstruktur. So werden zwei Kandidaten der gleichen Gruppe zugeordnet, wenn...:

- ...beide sich im gleichen Typ von Listen-Container (entweder in einer HTML-Tabellenzeile, oder in der gleichen Art von HTML-Liste) befinden
- (bei HTML-Listen) ... und beide an der gleichen Position der Links ihres HTML-Listenelementes sind (also z.B. die ersten Links in den HTML-Listenelementen einer gepunkteten Liste)
- (bei HTML-Tabellen) ...und beide in der gleichen Spalte der HTML-Tabelle sind

4.5.2. Bewertung der Gruppen

Die Gruppen werden nun durch die Summe der Bewertungen ihrer Kandidaten bewertet. Die Gruppe mit der höchsten Bewertung wird dann als Listengruppe angenommen und ihre Kandidaten als Listenelemente ausgewählt.

5. Evaluation der Listenerkennung

5.1. Überblick

Da die Listenerkennung die Grundlage für alle weiteren Schritte von WikiListSuggest ist, ist es sehr wichtig einschätzen zu können, wie gut die Erkennung funktioniert. Um die Qualität der Erkennung zu messen, führten wir eine Evaluation auf 34 von uns während der Entwicklung nicht benutzten Listen durch. Hauptergebnisse dieser Evaluation waren:

- Am besten erkannt werden Listenartikel, bei denen sich an einer Stelle in einer HTML-Liste oder in einer Spalte einer HTML-Tabelle alle Listenelemente und sonst nichts befinden.
- Manchmal schlecht erkannt werden Listenartikel, bei denen sich die Listenelemente in einer HTML-Liste befinden, die HTML-Liste aber auch andere Links enthält
- Sehr problematisch sind Listenartikel, bei denen viele der Listenelemente keine Wikipedia-Links haben
- In seltenen Fällen gibt es Probleme mit der Struktur von HTML-Tabellen

Im folgenden wird zunächst die Methodik der Evaluation erklärt und dann genauer auf die Ergebnisse eingegangen.

5.2. Methodik

Bei der Evaluation wollten wir sicherstellen, dass unsere Testlisten repräsentativ für die Listen auf Wikipedia sind und nicht auf irgendwelche Weise spezielle Eigenschaften haben. Dazu benutzten wir folgende Auswahlmethodik:

1. Gehe auf die Wikipedia-Seite, auf der Listen kategorisiert erfasst werden.¹
2. Gehe jede Unterkategorie auf dieser Seite durch: Falls der oberste Artikel dieser Unterkategorie eine Liste ist, wähle diesen als Testliste aus. Wenn der oberste Artikel eine neue Unterkategorie ist, öffne diese Unterkategorie und versuche das gleiche erneut, bis eine Testliste gefunden wurde.

Insgesamt kamen wir so auf eine Menge von 34 Testlisten aus 34 verschiedenen Kategorien (siehe Abschnitt A.1).

¹<http://en.wikipedia.org/wiki/Category:Lists>

5.3. Ergebnisse

Die 34 Listen lassen sich bei der Erkennung in 5 Kategorien einteilen:

Ergebniskategorie	HTML-Listen	HTML-Tabellen	Gesamt
Perfekt erkannt	10	6	16
Zuviel erkannt	5	0	5
Teilweise erkannt	2	1	3
Nicht erkannt	2	2	4
Schwer erkennbar	1	4	6*

*eine der schwer erkennbaren Listen befand sich weder in einer HTML-Liste noch in einer Tabelle, sondern in einem Fließtext.

Tabelle 5.1.: Evaluationsergebnisse

Die Kategorien sind folgendermaßen definiert:

- Perfekt erkannt: Alle Listenelemente wurden erkannt, keine zusätzlichen Elemente wurden fälschlicherweise erkannt
- Zuviel erkannt: Alle Listenelemente wurden erkannt, einige zusätzliche Elemente wurden fälschlicherweise erkannt
- Teilweise erkannt: Die Listenelemente wurden teilweise erkannt, evtl. gab es auch falsch erkannte Elemente
- Nicht erkannt: Listenelemente hätten erkannt werden können, wurden jedoch gar nicht erkannt.
- Schwer erkennbar: Listenelemente konnten von WikiListSuggest nicht erkannt werden, da es überhaupt keine Listenelemente mit Wikipedia-Artikeln gab.

Überhaupt für WikiListSuggest erkennbar waren also nur 28 der 34 Listen. Listen ohne Wikipedia-Artikel bestanden hauptsächlich aus 2 Typen von Listenelementen. Erstens Listenelemente, die man nicht durch einen Namen beschreiben kann, wie bei der Liste Berliner Mauerstücke in Abbildung 5.1. Zweitens Listenelemente, die einen Namen, aber keinen Wikipedia-Artikel haben, wie bei der Liste der *ABS-CBN-Firmenslogans*.

Insgesamt wurden also 6 der 10 erkennbaren Listen in Tabellen erkannt und 10 der 18 erkennbaren Listen in HTML-Listen. Nutzbar für die Querygeneration waren 21 der 28 erkennbaren Listen: Alle komplett erkannten Listen, sowohl die ohne als auch die mit zusätzlich fälschlicherweise erkannten Listenelementen. Bei den teilweise erkannten Listen lag die Erkennung im Schnitt bei 70% der Listenelemente.

In Europe	[edit]
Freedom Park, Berlin	[edit]
A growing collection of painted sections of the wall are on permanent display in Stralauer Allee 3 in Berlin. ^[1]	
Le Quai Theatre, Angers, France	[edit]
A segment of the wall is on display inside Le Quai contemporary theatre in Angers, France . It will take a central part in the 2011 edition of "Les Accroche-coeurs" festival. ^[2]	

Abbildung 5.1.: Auszug aus der Liste Berliner Mauerstücke

5.4. Gründe für Erkennung oder Nichterkennung

Bei Listen, die komplett erkannt wurden, mussten sich alle Listenelemente in einer von WikiListSuggest erkennbaren Struktur befinden:

- Alle Listenelemente in einer Tabellenspalte
- Alle Listenelemente in einer HTML-Liste, und immer an der gleichen Position innerhalb jedes HTML-Listenelementes

Listen mit fälschlicherweise zusätzlich erkannten Elementen waren ohne Ausnahme Listen, die sich in HTML-Listen befanden. Die falsche Erkennung lag meist daran, dass bei manchen HTML-Listenelementen kein Link auf ein Listenelement war. Für diese HTML-Listenelemente wurde dann ein anderer Link fälschlicherweise als Listenelement ausgewählt. Ein Beispiel hierfür ist in Abbildung 5.2 zu sehen. Bei den teilweise erkannten Listen gab es drei verschiedene Probleme:

- HTML-Liste, bei der ein HTML-Listenelement mehrere Listenelemente enthielt
- HTML-Liste, bei der bei sehr wenigen Listenelementen das echte Listenelement manchmal der dritte statt der zweite Link war
- Tabelle mit einer HTML-Struktur, bei der die erste Tabellenzeile eine Spalte mehr enthält und dadurch alle Spalten in dieser Zeile um eins verschoben waren.

Bei gar nicht erkannten Listen wurde von WikiListSuggest eine falsche Struktur als Listenstruktur identifiziert. Dies passierte nur bei Listen mit sehr wenigen erkennbaren Listenelementen, insbesondere wenn eine andere Struktur wesentlich mehr Wikipedia-Links enthielt als die tatsächliche Listenstruktur. Ein Beispiel dafür ist die in Abbildung 5.3 gezeigte Liste von Kandidaten, die in der TV-Serie *Amazing Race Asia* angetreten sind. Problematisch war es hierbei auch, wenn das Themenwort des Titels nicht korrekt mit den Oberbegriffen der Listenelemente verglichen wurde: Dies konnte einerseits passieren, wenn der Oberbegriff zwar ein Synonym des Themenworts war,

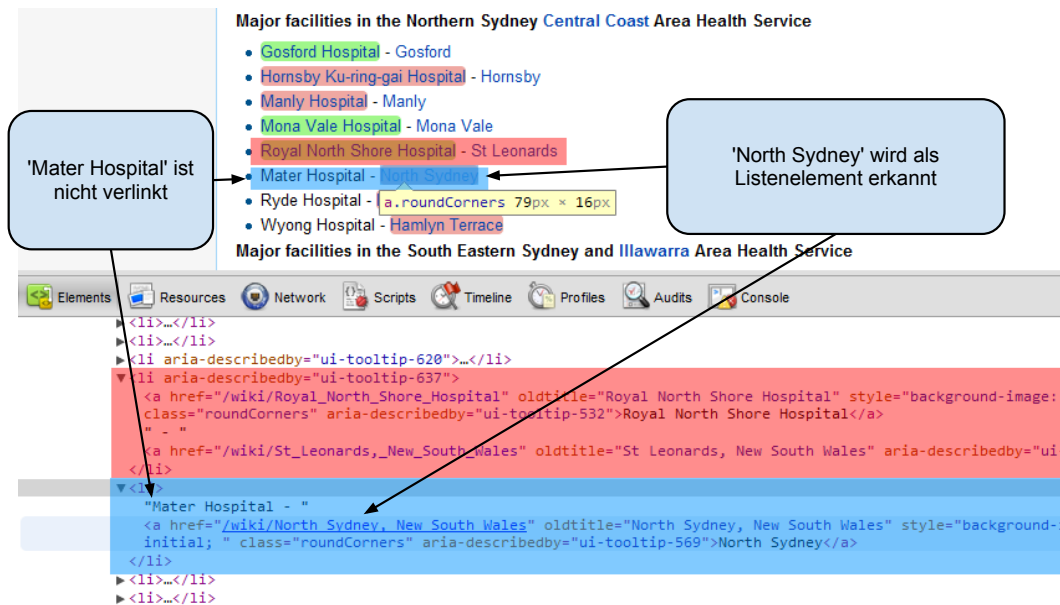


Bild von Simon Skilevic[Ski12].

Abbildung 5.2.: Auszug aus der Liste australischer Krankenhäuser

Contestants

The teams are arranged based on their placings and the team members are arranged based on the op

Name	Occupation	Age	Hometown
Ernesto "Ernie" Lopez	Businessman	42	Manila, Philippines
Jill "Jeena" Lopez	Homemaker	35	Manila, Philippines
Jacqueline Yu	Model	23	Manila, Philippines
Aubrey Miles	Actress	26	Manila, Philippines

Bild von Simon Skilevic[Ski12].

Abbildung 5.3.: Auszug aus der Liste der Amazing Asia Race Contestants

aber dies in unserer Synonymdatenbank nicht vermerkt war. Andererseits konnte es auch aus anderen sprachlichen Gründen Probleme beim Vergleich geben: Bei einer Liste war *amphitheater* der Oberbegriff der Listenelemente, das Themenwort der Liste war allerdings *amphitheatre*. Dadurch wurden die richtigen Listenelemente nicht hoch genug bewertet, um erkannt zu werden.

Drei seltene Probleme, die bei unserer Arbeit, aber nicht in der Evaluation auftauchen, waren:

- Listen mit Listenelementen in 2 verschiedenen Spalten von der selben HTML-Tabelle
- Listen mit Listenelementen in Tabellen *und* HTML-Listen
- Listen mit Listenelementen, die gar nicht in HTML-Listen oder Tabellen waren

6. Zukünftige Arbeit - Listenerkennung

6.1. Zukünftige Arbeit

Aus der Evaluation lassen sich mehrere Ansätze für zukünftige Verbesserungen ableiten:

- Bei HTML-Listen genauere Betrachtung der einzelnen HTML-Listenelemente
- Bei kleinen Listen Beachtung der Bewertung der HTML-Strukturen im Durchschnitt, also pro Kandidat
- Bei HTML-Tabellen anpassen der HTML-Tabellen-Struktur, falls Spaltenanzahl nicht in jeder Zeile gleich
- Beachtung anderer Schreibweisen sowie weiterer Synonyme beim Themenwort

6.2. Verbesserungen bei HTML-Listenelementen

Um falsche Treffer bei HTML-Listenelementen auszuschließen, könnte man hauptsächlich zwei Faktoren einbeziehen:

- Ob der Kandidat zu anderen Oberbegriffen gehört als der Rest der Liste
- Ob vor dem Kandidat im HTML-Listenelement viel Text steht

Beide Faktoren machen es unwahrscheinlich, dass ein Kandidat Teil der Liste ist, selbst wenn er in der gleichen Struktur wie der Rest der Liste ist.

Eine Schwierigkeit wäre hierbei die genaue Implementierung: Wie unterschiedlich die Oberbegriffe eines Kandidaten im Vergleich zur restlichen Liste sein müssen bzw. wie viel Text vor einem Kandidaten stehen muss, um ihn sicher ausschließen zu können.

Auch für die fehlende Erkennung bei HTML-Listen können die Oberbegriffe der Kandidaten hilfreich sein. WikiListSuggest könnte überprüfen, ob die nicht als Listenelemente erkannten Kandidaten im HTML-Listenelement zu den gleichen Oberbegriffen gehören wie die erkannte Liste. Es lässt sich auch beides kombinieren: Wenn

ein erkannter Kandidat von den Oberbegriffen her nicht zum Rest der Liste passt, ein nicht erkannter Kandidat im gleichen Listenelement aber schon, könnte man ihn stattdessen auswählen. Wieder muss aufgepasst werden, ab wann genau man andere Kandidaten als den zur erkannten Struktur gehörenden in die Liste aufnimmt, da die Zuordnung der Kandidaten zu den Oberbegriffen nicht immer perfekt ist.

6.3. Verbesserungen bei kleinen Listen

Kleine Listen werden manchmal nicht richtig erkannt, weil eine falsche Struktur viel mehr Kandidaten als die richtige enthält und allein dadurch eine höhere Bewertung erreicht. Hier könnte man versuchen, beim Vergleich zwischen ungleich großen Strukturen sowohl die Summe der Bewertungen ihrer Kandidaten einzubeziehen als auch den Durchschnitt. Wenn der Durchschnitt einer kleineren Struktur wesentlich höher ist als bei einer größeren, könnte man sie als Listenstruktur auswählen.

6.4. Verbesserungen bei Tabellen

Bei Tabellen könnte man die Anzahl der Spalten betrachten: Wenn diese in einer Zeile von den anderen Zeilen abweicht, könnte man in dieser Zeile z.B. die erste Spalte ignorieren. Allerdings müssten für solche Verbesserungen die Struktur von Tabellen auf Wikipedia-Listen-Artikeln noch genauer untersucht werden, um Anpassungen der Tabellenstrukturen zu finden, die tatsächlich eine Verbesserung gegenüber gar keiner Anpassung darstellen.

6.5. Verbesserung beim Themenwort-Vergleich

Für den Vergleich von Oberbegriffen mit dem Themenwort wären Vergleichsfunktionen, die sprachliche Besonderheiten beachten, nützlich. So könnte verhindert werden, dass z.B. verschiedene Schreibweisen im britischen und amerikanischen Englisch (*theatre*, *theater*) zu Vergleichsfehlern führen.

Auch eine besser für unsere Aufgaben geeignete Synonymdatenbank könnte eventuell beim Vergleich zwischen Oberbegriff und Themenwort weiterhelfen.

6.5.1. Potential der verschiedenen Ansätze

Bezogen auf unsere Evaluationslisten wären die Verbesserungen bei HTML-Listenelementen bei guter Umsetzung am hilfreichsten: Die fünf Listen, bei denen neben der richtigen

Liste auch falsche Elemente als Listenelemente erkannt wurden, könnten dann komplett richtig erkannt werden. Des Weiteren könnten zwei der nur teilweise erkannte Listen komplett erkannt werden.

Durch die Verbesserungen für kleine Listen würde die Erkennung bei zwei der gar nicht erkannten Listen verbessert: Bei einer Liste ist es der einzige erfolgversprechende Ansatz der genannten Ansätze, bei der anderen könnte auch die Verbesserung des Themenwort-Vergleichs noch helfen.

Die Tabellen-Verbesserungen helfen nur bei einer der teilweise erkannten Listen. Sie sind auch eher schwer zu implementieren.

Die Verbesserung beim Themenwort-Vergleich könnten bei zwei der gar nicht erkannten Listen zu einer korrekten Erkennung führen.

7. Benutzeroberfläche

7.1. Funktionalitäten

Damit WikiListSuggest tatsächlich zur Erweiterung von Wikipedia-Listen benutzt werden kann, muss die Benutzeroberfläche für diesen Zweck gestaltet sein. Sie muss daher dem Nutzer folgendes ermöglichen:

- die Korrektheit der Listenerkennung überprüfen
- die Qualität der Anfrage von Broccoli einschätzen
- die Anfrage an Broccoli verändern
- die Erweiterungsvorschläge übersichtlich sehen
- die Erweiterungsvorschläge schnell auf ihre Richtigkeit hin überprüfen

Dies alles sollte möglich sein, ohne die Wikipedia-Seite zu verlassen. Zur Veranschaulichung dieser Funktionalitäten gehen wir einen typischen Ablauf eines Nutzers, der eine Liste erweitern will, durch.

7.2. Typischer Use Case

Ein typischer Nutzerablauf sieht wie folgt aus:

1. **Einbindung und Start:** Binde WikiListSuggest per Bookmarklet in die Wikipedia-Seite ein und starte die Erkennung
2. **Qualitätscheck:** Überprüfe Listenerkennung und vorgeschlagene Broccoli-Anfrage
3. (optional) **Anfrageverbesserung:** Verbessere manuell die Broccoli-Anfrage
4. **Überprüfung der zusätzlichen Elemente:** Prüfe, ob zusätzliche Elemente aus Broccoli-Anfrage in die Liste passen, wiederhole evtl. Schritt 3

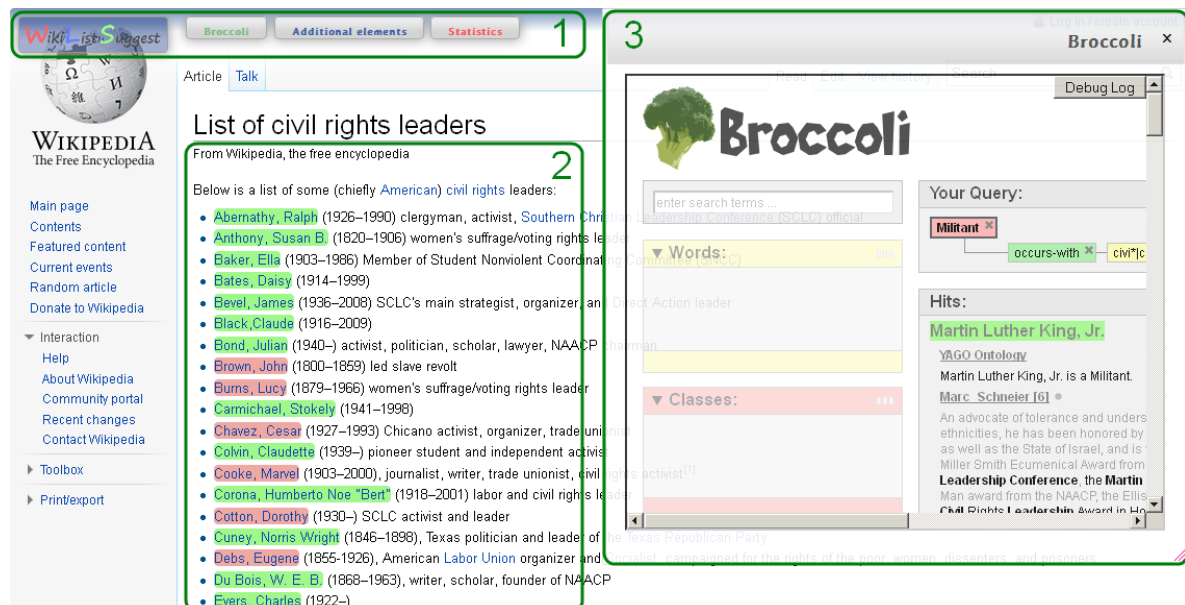
7.3. Einbindung und Start

Für die Einbindung von WikiListSuggest in eine Wikipedia-Seite benutzen wir *Bookmarklets*. Bookmarklets sind Lesezeichen im Browser, in denen keine URL gespeichert ist, sondern Javascript-Code. In unserem Fall lädt dieser Javascript-Code

WikiListSuggest in die Seite hinein. Nach der Einbindung in die Seite wird unser Programm über den Start-Button gestartet.

7.4. Qualitätscheck

Nachdem der Startvorgang beendet ist, werden dem Nutzer die nötigen Informationen gezeigt, um die Korrektheit der Listenerkennung und die Qualität der Anfrage beurteilen zu können. Gleichzeitig wird dem Nutzer bereits das Menü angezeigt, was er für zukünftige Schritte benötigt. Dies kann man in Abbildung 7.1 sehen: Die erkannten Listenelemente sind farbig markiert, grün, wenn Broccoli sie auch gefunden hat, rot, wenn Broccoli sie nicht gefunden hat. Gleichzeitig ist das Broccoli-Interface bereits geöffnet, so dass der Nutzer die Anfrage sehen kann. So kann der Nutzer sowohl sehen, ob die Liste korrekt erkannt wurde, als auch, ob die vorgeschlagene Anfrage an Broccoli plausibel ist.

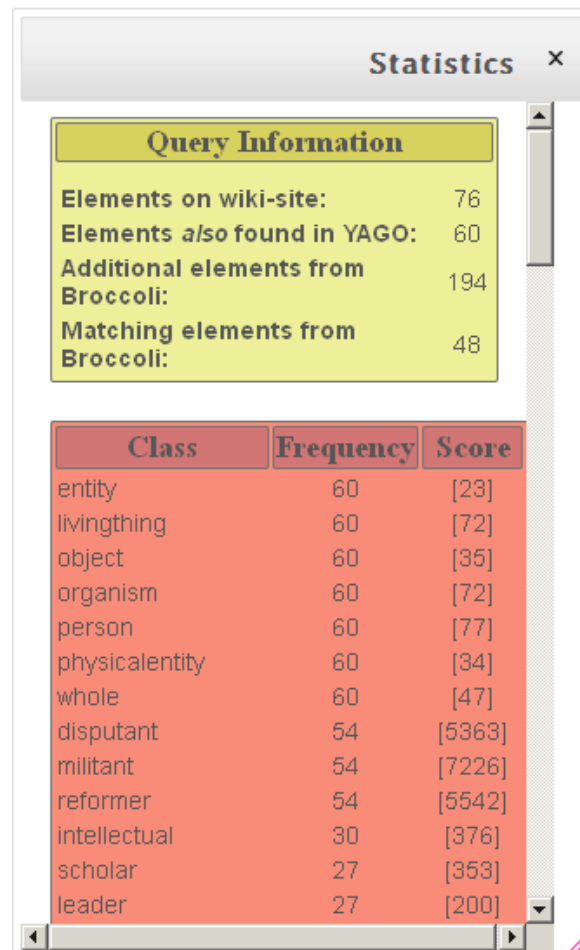


- 1 WikiListSuggestMenü mit Buttons zum Öffnen der Fenster: Broccoli (Broccoli-Oberfläche), Additional Elements (zusätzliche von Broccoli gefundene Elemente) und Statistics (Statistiken über die Listenelemente und die Broccoli-Anfrage)
- 2 Erkannte Listenelemente, grün, wenn auch von Broccoli-Anfrage gefunden, rot, wenn nicht
- 3 Broccoli-Oberfläche mit vorgeschlagener Anfrage

Abbildung 7.1.: WikiListSuggest-Oberfläche nach dem Startvorgang

7.5. Anfrageverbesserung

Als nächstes kann der Nutzer versuchen, die Broccoli-Anfrage zu verbessern, damit das Suchergebnis näher an die “echte” Liste herankommt. Dabei hilft das Statistik-Fenster (Abbildung 7.2).



The screenshot shows a window titled 'Statistics' with a close button. It contains two main sections: 'Query Information' and a table of class frequencies and scores.

Query Information	
Elements on wiki-site:	76
Elements also found in YAGO:	60
Additional elements from Broccoli:	194
Matching elements from Broccoli:	48

Class	Frequency	Score
entity	60	[23]
livingthing	60	[72]
object	60	[35]
organism	60	[72]
person	60	[77]
physicalentity	60	[34]
whole	60	[47]
disputant	54	[5363]
militant	54	[7226]
reformer	54	[5542]
intellectual	30	[376]
scholar	27	[353]
leader	27	[200]

In der gelben Box sind vier nützliche Zahlen: Wie viele Listenelemente es auf der Wikipedia-Seite gibt, wie viele Elemente auch YAGO bekannt sind, wie viele zusätzliche Elemente und wie viele gleiche Elemente von der Broccoli-Anfrage gefunden wurden.

In der roten Box sind die Oberbegriffe der Listenelemente mit ihrer Anzahl und ihrer Bewertung durch WikiListSuggest

Abbildung 7.2.: Statistik-Fenster für die Liste der Bürgerrechtler

In ihm kann man sehen, wie viele der Listenelemente im Suchergebnis der Broccoli-Anfrage enthalten sind und wie viele zusätzliche Entitäten gefunden werden. Außerdem geben die Oberbegriffe der Listenelemente Hinweise, welche Oberbegriffe bei der Anfrage an Broccoli benutzt werden können.

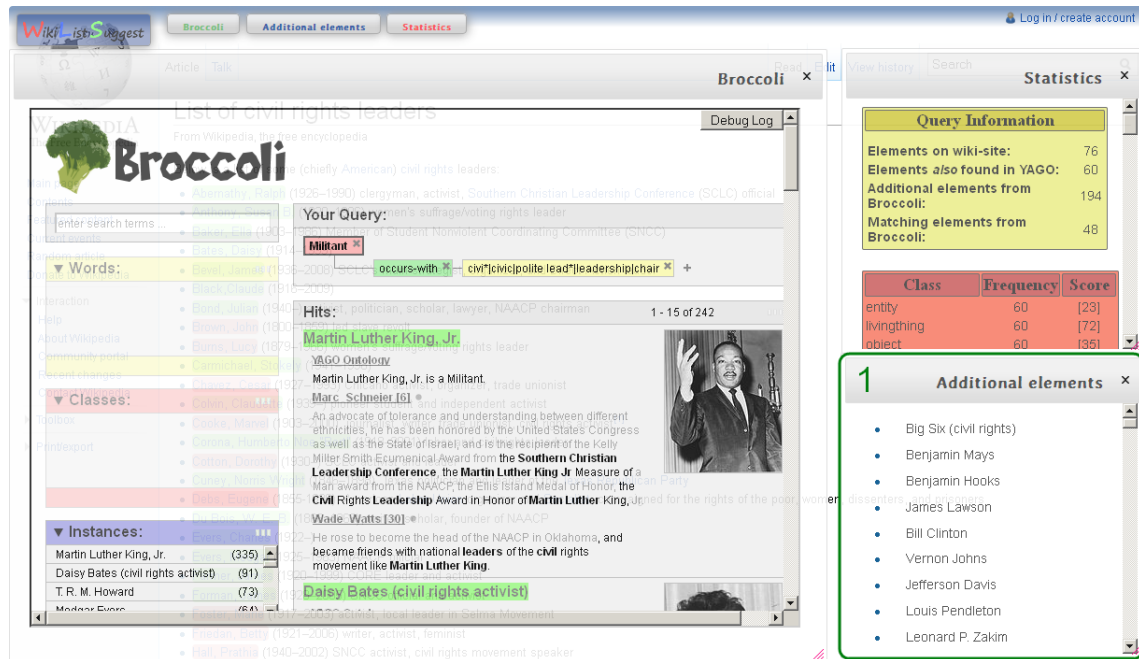


Abbildung 7.3.: Gute Anordnung der Fenster für die Anfrageverbesserung

Abbildung 7.3 zeigt eine für die Anfragabeverbesserung besonders bequeme Anordnung der Fenster. Alle drei Fenster sind sichtbar: Links die Broccoli-Oberfläche, rechts oben das Statistik-Fenster, rechts unten das Fenster mit zusätzlichen Elementen der Broccoli-Anfrage. Mit diesen drei Fenstern kann man in einem schnellen Rhythmus - Anfrageveränderung, Prüfen der Anzahl überdeckter und zusätzlicher Elemente, Anfrageveränderung, ... - versuchen, bessere Anfragen zu finden. Bei jeder Änderung der Broccoli-Anfrage aktualisieren sich die anderen beiden Fenster automatisch.

7.6. Überprüfung der zusätzlichen Elemente

Zum Überprüfen der Vorschläge hat der Nutzer zwei Möglichkeiten: In Broccoli selbst sind die in der Wikipedia-Liste enthaltenen Entitäten grün und die zusätzlichen Entitäten blau markiert. Zusätzlich bietet unsere Benutzeroberfläche eine Liste, in der nur die zusätzlichen Elemente des Broccoli-Ergebnisses angezeigt werden. Wenn man mit der Maus auf eines der Elemente fährt, erscheint ein Popup mit den Informationen aus dem Broccoli-Suchergebnis zu diesem Element:

- Welchen Oberbegriff das Element in YAGO hat
- Textauszüge aus Wikipedia, in dem das gefundene Element zusammen mit den gesuchten Wörtern vorkommt, sowie die Artikel, aus denen die Textauszüge stammen

Dieses Fenster ist besonders nützlich, wenn sehr viele Elemente aus der Wikipedia-Liste in der Broccoli-Liste sind, so dass die zusätzlichen Elemente in der Broccoli-Oberfläche weit verstreut sind. Der Nutzer kann auch Elemente aus dieser Liste löschen, wenn er sie als falsch erkannt hat.

Die zusätzlichen Elemente, die in die Wikipedia-Liste übernommen werden sollen, kann sich der Nutzer dann durch ein handelsübliches Programm wie Notepad oder durch Zwischenspeichern in der Adressleiste eines anderen Tabs des Browsers merken :)

7.7. Anpassbarkeit der Oberfläche

Unsere Oberfläche lässt sich durch den Nutzer sehr gut an seine Bedürfnisse anpassen:

- Alle Fenster kann man beliebig verschieben und vergrößern/verkleinern
- Alle Fenster kann man schließen und wiederaufmachen, sie merken sich dabei ihre vorherige Position und Größe
- Alle Fenster kann man per Doppelklick auf die Größe der kompletten Seite vergrößern und per erneutem Doppelklick auf ihre ursprüngliche Größe und Position wiederherstellen.
- Die komplette Benutzeroberfläche lässt sich durch den Klick auf das WikiListSuggest-Logo ein- und ausblenden

8. Benutzeroberfläche - Implementierung

8.1. Überblick

Die Oberfläche von WikiListSuggest muss im Browser mit zwei unabhängigen Systemen umgehen:

- Der Wikipedia-Seite selbst
- Der Broccoli-Oberfläche

Zusätzlich muss sie im Hintergrund mit unserem WikiListSuggest-Backend und mit dem Broccoli-Backend kommunizieren können.

Wir benutzen dafür vier Technologien:

- **Bookmarklet:** *Bookmarklets* sind einfache Lesezeichen im Browser, in denen statt einer URL Javascript-Code steht. Wenn man auf das Bookmarklet klickt, wird man also nicht von der aktuellen Seite weggeführt. Stattdessen wird der im Bookmarklet gespeicherte Javascript-Code auf der aktuellen Seite ausgeführt. Oft wird ein Bookmarklet benutzt, um mit diesem weitere Javascript-Code-Files nachzuladen. Dies ist auch deswegen nützlich, da Lesezeichen in den Browsern meist nur eine begrenzte Länge haben dürfen.
- **JSONP:** *JSONP* ist eine Javascript-Technologie, mit der die Kommunikation von einer Seite (in unserem Fall die Wikipedia-Listen-Seite) zu einer Internet-Adresse auf einer anderen Domain möglich wird. Eine normale HTTP-Anfrage durch JavaScript ist hierbei aus Sicherheitsgründen nicht möglich. Es ist aber möglich, ein neues JavaScript-Element in die Seite hinzuzufügen. Dessen Quell-URL kann im Namen eine "versteckte" Anfrage enthalten. Der angefragte Server muss dann mit ausführbarem Javascript-Code antworten, der bei Beendigung der Anfrage direkt im Browser ausgeführt wird. Normalerweise wird dem Server bei der Anfrage hierbei ein Funktionsname mitgeschickt, den er mit einem Antwort-Objekt in JSON-Notation aufrufen soll.
- **postMessage:** *postMessage* ist eine neuere Javascript-Funktion, die die Kommunikation zwischen verschiedenen Frames, die auf der gleichen Seite sind, aber von verschiedenen Domains stammen, ermöglicht. Hierbei muss der Sender eine Nachricht an den Zielframe schicken. Der Zielframe muss dabei selbst aktiv auf diese Nachrichten warten. Sowohl Sender als auch Empfänger können

hierbei zur Sicherheit nur Nachrichten an bestimmte Domains senden bzw. empfangen.

- **URL-Adresse:** Zusätzlich zur Kommunikation per `postMessage` kommunizieren wir mit der Broccoli-Oberfläche auch mithilfe der URL-Adresse, mit der wir den Frame öffnen. In die URL kann bereits eine Anfrage geschrieben werden, die die Oberfläche dann direkt an das Broccoli-Backend schickt.

Im folgenden wird die technische Architektur, die unter anderem diese Technologien benutzt, vorgestellt. Sie wird beschrieben, indem wir die Hintergrundaktivitäten beim Use Case von oben erläutern.

8.2. Einbindung und Start

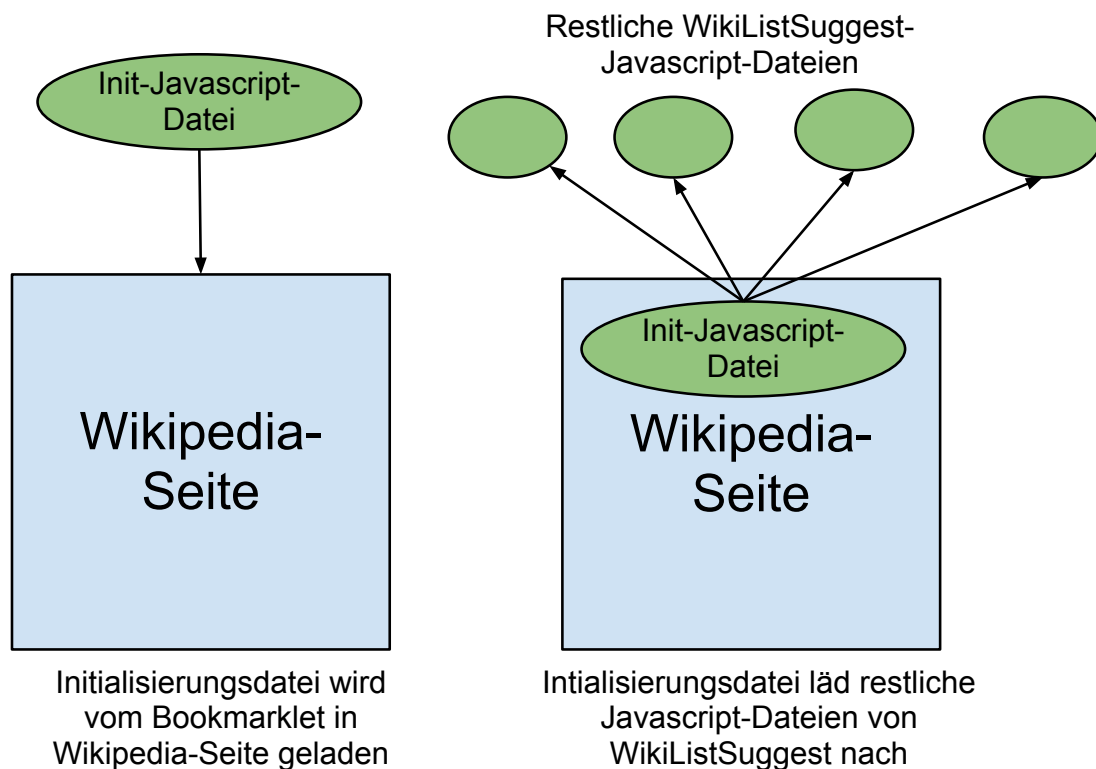
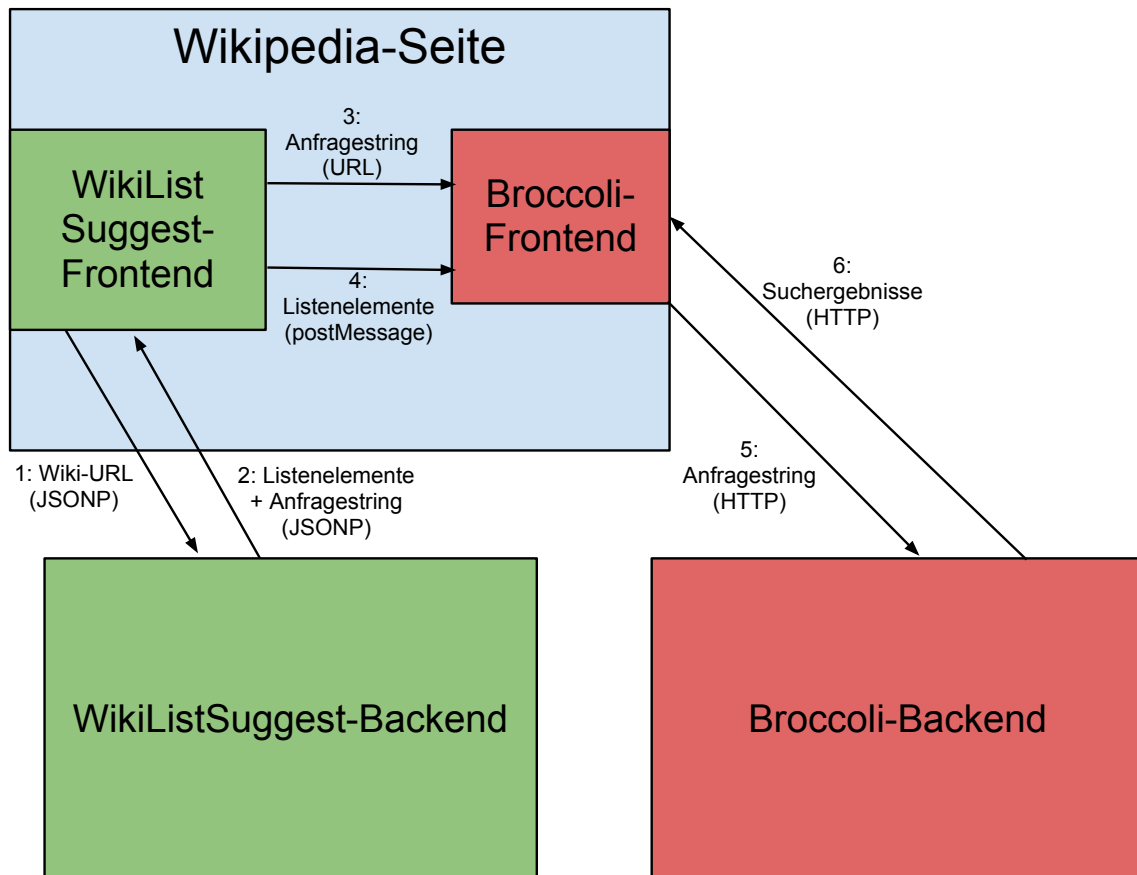


Abbildung 8.1.: Initialisierungsphase von WikiListSuggest

Zuerst wird über ein Bookmarklet ein Script in die Wikipedia-Seite eingefügt, was den Rest unserer WikiListSuggest-Javascript-Dateien nachlädt.

Beim Klicken auf den Start-Button schickt unsere Oberfläche per JSONP die URL der Wikipedia-Seite an unserem WikiListSuggest-Server. Dieser erkennt die Listenelemente auf der Seite und sendet per normalen HTTP-Requests verschiedene

Anfragen an das Broccoli-Backend. Die erkannte Liste und die beste Anfrage wird dann an unsere WikiListSuggest-Oberfläche zurückgesendet.

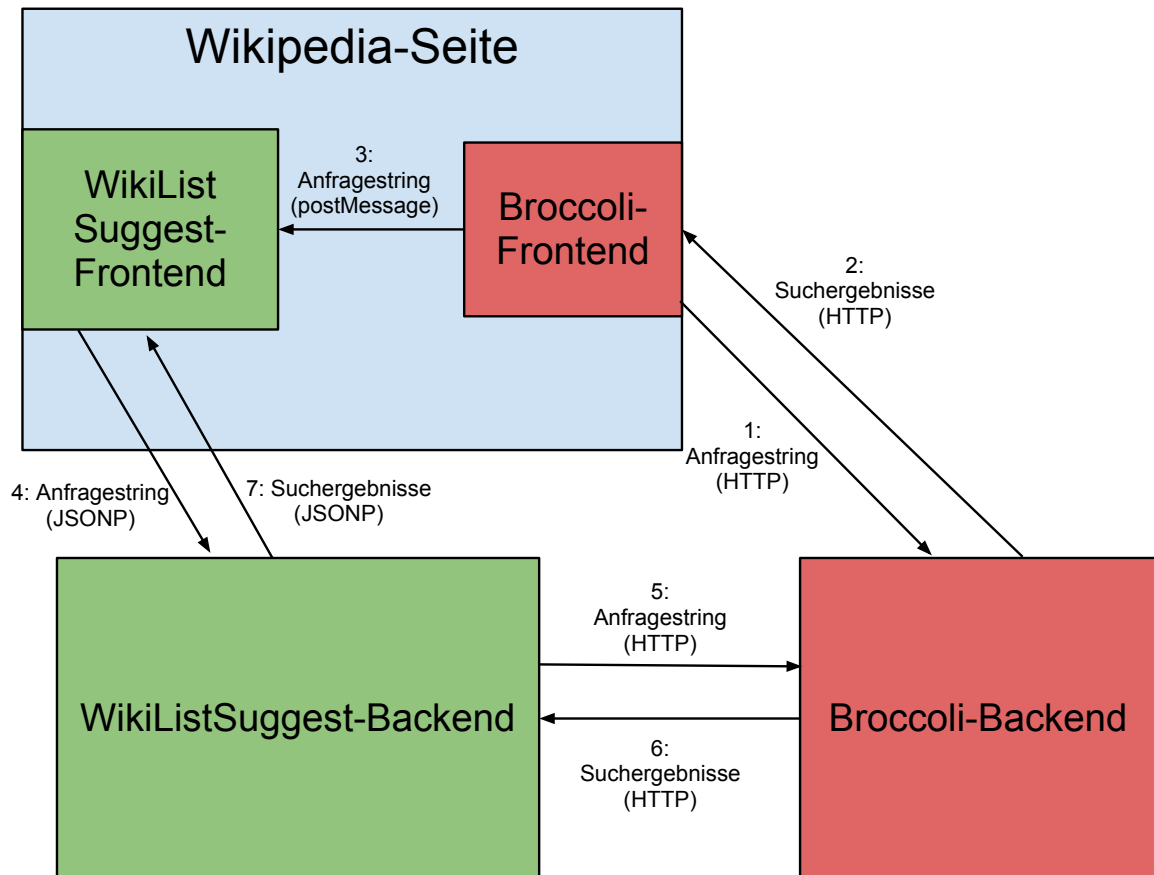


Zwischen Schritt 3 und 4 meldet das Broccoli-Frontend noch per `postMessage`, dass es geladen ist, dies wurde aus Übersichtlichkeitsgründen weggelassen

Abbildung 8.2.: Startphase von WikiListSuggest

Nun beginnt die Kommunikation zwischen unserer Oberfläche und dem Broccoli-Frame. Da sie aufgrund von Sicherheitsbeschränkungen nicht direkt aufeinander zugreifen können, benutzen wir die `postMessage`-Funktion und die URL-Adresse des Broccoli-Frames für die Kommunikation. Sobald die Antwort unseres WikiListSuggest-Backends angekommen ist, wird der Broccoli-Frame geöffnet. Dabei wird ihm über die URL-Adresse die vom WikiListSuggest-Backend vorgeschlagene Anfrage mitgeteilt. Der Broccoli-Frame selbst bekommt die Suchergebnisse dann direkt vom Broccoli-Backend. An den Broccoli-Frame werden außerdem zusätzlich, sobald er geladen ist, die erkannten Listenelemente per `postMessage` geschickt (der Broccoli-Frame meldet, dass er geladen ist, auch per `postMessage`). So kann der Broccoli-Frame markieren, welche seiner Treffer auch auf der Wikipedia-Liste sind. Die Kommunikation bis hierhin kann man in Abbildung 8.2 nachverfolgen.

Der Broccoli-Frame schickt danach wiederum den Anfragestring seiner Anfrage (immer noch die von WikiListSuggest vorgeschlagene) für das Broccoli-Backend zurück. Unsere Oberfläche fragt dann über das WikiListSuggest-Backend das Broccoli-Backend nach den Entitäten für die ausgewählte Anfrage. Der letzte Schritt wäre eigentlich nicht notwendig, da das WikiListSuggest-Backend bereits beim Auswählen der besten Anfrage die Anfrage an Broccoli geschickt hat. Er wirkt sich allerdings auch nicht besonders schlimm aus, da das Broccoli-Backend die Ergebnisse der letzten Anfragen zwischenspeichert. Die gesamte Kommunikation für die Anfrage der Broccoli-Ergebnisse ist in Abbildung 8.3 dargestellt.



Eigentlich laufen Schritt 1 und 3 gleichzeitig ab und die nachfolgenden Schritte 2 sowie 4,5,6,7 parallel. Zur Vereinfachung wurde dies hier stattdessen sequentiell dargestellt.

Abbildung 8.3.: Schicken der Broccoli-Anfragen von WikiListSuggest

8.3. Qualitätscheck

Für den ersten Qualitätscheck müssen die Elemente auf der Seite rot markiert werden, die erkannt wurden, und die grün markiert werden, die auch auf Broccoli gefun-

den wurden. Da unsere Javascript-Dateien zu Beginn direkt in die Wikipedia-Seite eingefügt wurden, kann unsere Oberfläche hier auf die Elemente der Wikipedia-Seite zugreifen. Der Vergleich zwischen den Elementen der Wikipedia-Liste und der Broccoli-Liste erfolgt dabei über die Links der Elemente: Wenn sie gleich sind, werden die Elemente als gleich betrachtet. Hierbei werden auch sogenannte redirects beachtet: Wenn ein Wikipedia-Artikel-Link zwar anders geschrieben ist, aber auf die gleiche Seite verweist, wird er auch als gleich erkannt.

8.4. Anfrageverbesserung

Bei der Anfrageverbesserung ändern wir die Anfrage zunächst in dem Broccoli-Frame. Dies passiert unabhängig von WikiListSuggest, für eine Beschreibung der Architektur von Broccoli siehe [Bä11]. Sobald die Anfrage geändert wurde, passiert das gleiche wie am Anfang:

Der Broccoli-Frame schickt den dazugehörigen Anfragestring per `postMessage` an unsere WikiListSuggest-Oberfläche. Diese schickt dann über unser WikiListSuggest-Backend eine Anfrage an das Broccoli-Backend und bekommt das Ergebnis der Broccoli-Anfrage zurück. Wieder gilt, dass die Anfrage doppelt gestellt wird, vom Broccoli-Frame und von der WikiListSuggest-Oberfläche, allerdings ist dies wegen des Cachings von Broccoli nicht sehr kostspielig. Die Entscheidung für diese Lösung wurde hauptsächlich getroffen, damit der Code der Broccoli-Oberfläche nur minimal geändert werden muss. So können wir auch bei zukünftigen Veränderungen in der Broccoli-Oberfläche unsere Erweiterungen schnell wieder einbauen.

8.5. Überprüfen zusätzlicher Elemente

Beim Überprüfen der zusätzlichen Elemente durch den Nutzer werden die Informationen von Broccoli über diese Elemente benötigt. Dafür schickt unsere Oberfläche wiederum über das WikiListSuggest-Backend Anfragen an Broccoli für das jeweilige Element und dies erst, wenn über ein Element mit der Maus gefahren wird. So verhindern wir lange Wartezeiten beim Laden der Liste.

8.6. Zukünftige Arbeit

Wie in der vorherigen Veranschaulichung erklärt, werden die Anfragen an Broccoli immer doppelt, am Anfang sogar dreimal gestellt. Die dreimalige Anfrage zu Beginn ließe sich ohne Änderungen am Code der Broccoli-Oberfläche auf zwei Anfragen reduzieren. Bei den späteren Anfragen ließe sich die zweite Anfrage evtl. auch verhindern, in dem wir die Broccoli-Oberfläche verändern. Diese könnte eine

vom Nutzer gestellte Anfrage gleich so an das Backend stellen, dass sie die nötigen Informationen vom Broccoli-Backend holt und diese per `postMessage` an die `WikiListSuggest`-Oberfläche schickt.

Eine noch nicht genannte Beschränkung unserer Oberfläche ist es, dass wir bei einer Anfrage an Broccoli nur die ersten 4000 Elemente betrachten. Wir tun dies aus Performancegründen: Eine größere Anfrage würde die Oberfläche lange blockieren. Außerdem könnte sie evtl. auch nicht verarbeitet werden, wenn das Ergebnis zu groß wäre und nicht in den für die Verarbeitungsfunktion vorgesehenen Speicher passen würde. Für die Zukunft könnte man hierzu die Anfragen splitten, also zuerst die ersten 4000 Elemente holen, dann die nächsten 4000, etc.

9. Zusammenfassung und Schlussfolgerungen

9.1. Momentaner Stand von WikiListSuggest

In dieser Arbeit haben wir beschrieben, wie wir mit Hilfe der Ontologie YAGO Listen auf Wikipedia erkennen können und mit Hilfe der semantischen Suchmaschine Broccoli Vorschläge für die Erweiterung machen können.

Wir erkennen dabei bereits gut 75% der Listen auf Wikipedia zumindest so gut, dass wir dafür Erweiterungsvorschläge machen können. Unser Programm WikiListSuggest hat bereits eine benutzbare Oberfläche, mit der man relativ schnell und gut verschiedene Anfragen an Broccoli ausprobieren kann, um die Liste zu erweitern.

9.2. Ansatzpunkte für Verbesserungen

Es bieten sich auch einige Ansatzpunkte für die Weiterentwicklung sowohl der Erkennung als auch der Oberfläche.

9.2.1. Listenerkennung

Für die Erkennung wären diese Ansatzpunkte hauptsächlich:

1. Eine genauere Analyse der Artikelstruktur, insbesondere der HTML-Listenelemente. Hier könnte man auch mehrere in einem HTML-Listenelement enthaltene Kandidaten für die Liste in Betracht ziehen, statt generell immer nur einen zu nehmen. Der Aufwand für erste Tests dürfte hier nur einige Stunden bis wenige Tage dauern, der Aufwand für eine robuste und gute Lösung ist aber schwerer abzuschätzen
2. Eine Anpassung der Auswahl der Listenstruktur für kleine Listen: Eventuell durch Beachtung des Durchschnittswertes von HTML-Strukturen des Artikels. Dies könnte schneller getestet und realisiert werden, erste Tests wären in wenigen Stunden, eine erste robuste Lösung wäre in mehreren Tagen vorstellbar.

9.2.2. Benutzeroberfläche

Für die Benutzeroberfläche wären die Haupt-Verbesserungsmöglichkeiten:

1. Eine Speicherung der Liste der zusätzlich von Broccoli gefundenen Elemente im Browser. So könnte man als Nutzer auch zum Editieren der Wikipedia-Liste übergehen, ohne die neuen Elemente vorher selber zwischenspeichern zu müssen. Der Aufwand variiert hier je nach Implementierung: Eine eher unintuitive Lösung, bei der man zweimal das Bookmarklet klicken muss, einmal auf der Wikipedia-Listen-Seite und einmal auf der Editier-Seite, wäre wohl innerhalb weniger Stunden bis Tage zu realisieren. Eine robustere Lösung, inklusive Umstellung auf ein Browser-Plugin dürfte eher einige Tage bis 1-2 Wochen dauern.
2. Eine Umstellung der Architektur, so dass doppelte Anfragen an Broccoli vermieden werden. Sollte die zusätzliche Last für Broccoli hierbei allerdings nicht deutlich sinken, könnte die Umstellung auch zu aufwendig sein: Eine Umstellung würde wahrscheinlich eine stärkere Veränderung des Codes des Broccoli-Interfaces zur Folge haben mit entsprechend höherem Aufwand, wenn das Broccoli-Interface selbst verändert und weiterentwickelt wird.

9.3. Evaluation durch Nutzer

Zum jetzigen Zeitpunkt wäre sicher auch mehr Feedback von tatsächlichen Nutzern hilfreich:

- Entwickler von Broccoli, die das Interface benutzen können, um Broccoli zu testen und weiterzuentwickeln
- Wikipedia-Editoren, die ein Interesse haben, Listen auf Wikipedia zu vervollständigen

Feedback von diesen beiden Nutzergruppen könnte helfen, die nächstbesten Entwicklungsschritte für WikiListSuggest zu bestimmen.

Danksagung

Ich möchte mich bei mehreren Leuten für die Unterstützung bei dieser Arbeit bedanken.

Zuallererst bei Simon Skilevic, der in dieser intensiven und anstrengenden Arbeitsphase immer hilfreich zur Seite stand, selbst in den kompliziertesten Situationen.

Natürlich auch bei unserer Betreuerin Hannah Bast, die unseren engen und stressigen Zeitplan akzeptiert und uns jederzeit unterstützt hat, wenn wir die Zeit zum Fragen gefunden haben ;) Durch Ihre Tipps und Ratschläge bin ich mir sicher, dass ich jetzt eine gute Bachelorarbeit schreiben könnte :)

Dazu natürlich auch

Ausserdem möchte ich mich bei allen Korrektoren bedanken, die mich unterstützt haben bei Manuel Watter für seine beständige, sehr gute und freundliche Korrektur, bei Linda Kleist für ihre wie immer hochkompetenten Ratschläge und natürlich bei meiner Familie und auch bei Manuel Meyer für die bereitwillige Last-Minute-Korrektur am Abend vor der Abgabe.

A. Anhang

A.1. Evaluationslisten

Nummer	Name
WL1	List of current A&M Records artists
WL2	List of banks in Albania
WL3	List of aerospace engineers
WL4	List of films based on English-language comics
WL5	List of Royal Australian Air Force aircraft squadrons
WL6	List of female philosophers
WL7	List of atheist activists and educators
WL8	List of Albania international footballers born outside Albania
WL9	African Americans in the United States Congress
WL10	List of EN standards
WL11	List of the world's 100 worst invasive species
WL12	Chronological list of saints and blesseds in the 12th century
WL13	List of Disney theme park attractions
WL14	List of class action lawsuits
WL15	List of constellations
WL16	List of best-selling albums
WL17	List of awards and nominations received by John Abraham
WL18	List of archaeological sites by country
WL19	List of alternate history fiction
WL20	List of hospitals in Australia
WL21	List of accidents and disasters by death toll
WL22	List of European birds
WL23	List of astronauts by name
WL24	Abydos King List
WL25	List of Roman amphitheatres
WL26	List of The Amazing Race Asia contestants
WL27	List of firsts in aviation
WL28	List of unsolved problems in biology
WL29	List of Alpha Kappa Alpha Boulés
WL30	ISO 3166-2:CA
WL31	Dolch word list
WL32	List of ABS-CBN Corporation slogans
WL33	List of Berlin Wall segments
WL34	List of words in English containing all the vowels

Grüne Listen wurden perfekt erkannt, gelbe Listen komplett mit zusätzlichen Elementen, rosa Listen teilweise, rote Listen gar nicht und graue Listen waren nicht erkennbar
Bild von Simon Skilevic[Ski12].

Abbildung A.1.: Evaluationslisten

Literaturverzeichnis

- [BBBH12] BAST, Hannah ; BÄURLE, Florian ; BUCHHOLD, Björn ; HAUSSMANN, Elmar: Broccoli: Semantic Full-Text Search at your Fingertips. (2012). – submitted
- [BSV11] BALOG, Krisztian ; SERDYUKOV, Pavel ; DE VRIES, Arjen P.: Overview of the TREC 2011 Entity Track. In: *NIST Special Publication : TREC 2011* (2011). – online available at <http://krisztianbalog.com/files/trec2011-entity-overview-wn.pdf>
- [Bä11] BÄURLE, Florian: A User Interface for Semantic Full Text Search. (2011). – online available at <http://ad.informatik.uni-freiburg.de/files/UserInterfaceSemanticFullTextSearch>
- [Ski12] SKILEVIC, Simon: Automatische Vorschläge zur Vervollständigung von Wikipedia-Listen - Evaluation, Anfragegenerierung. (2012)

