Bachelorarbeit

Extended Question Answering on Freebase

Matthias Urban

Sommersemester 2016

Albert-Ludwigs-Universität Freiburg
Technische Fakultät
Lehrstuhl für Algorithmen und Datenstrukturen

Lehrstuhlinhaberin: Prof. Dr. Hannah Bast
Supervisor: Prof. Dr. Hannah Bast

Abgabe: 31.08.2016

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.

_____ _____

Ort, Datum Unterschrift

Kurzfassung

Diese Bachelorarbeit beschäftigt sich mit Erweiterungen an dem Question Answering System (dt. Fragebeantwortungssystem) Aqqu [2]. Aqqu ist in der Lage einfache, natürlichsprachliche Fragen in SPARQL-Abfragen für die Wissensdatenbank Freebase zu übersetzen. Wenn man diese SPARQL-Abfragen ausführt, erhält man die Antwort auf die entsprechende Frage. Bisher war Aqqu nicht in der Lage Superlativ-Fragen wie “Was ist das größte Land in Europa?” oder Temporal-Fragen wie “Wer war der erste Präsident der Vereinigten Staaten?” zu beantworten. Aqqu konnte außerdem nur Fragen beantworten, deren SPARQL-Abfrage nur aus einer (möglicherweise k-ären) Relation besteht. Deshalb konnten Fragen wie “Wer ist die Frau des Präsidenten der Vereinigten Staaten?” nicht beantwortet werden. Im Folgenden werden diese Probleme mit mehreren Erweiterungen an Aqqu gelöst, sodass Aqqu in der Lage sein wird Fragen dieser Typen zu beantworten.

Des Weiteren ist das Answer Type Matching (dt. Antworttyperkennung) von Aqqu sehr einfach gehalten. Es wurde nur das Fragewort betrachtet. Diese Bachelorarbeit stellt ein verbessertes Answer Type Matching vor. Dieses basiert auf Machine Learning und betrachtet eine Reihe verschiedener Features (dt. Merkmale). Es funktioniert mit den Typen von Freebase und ist in der Lage die Antworttypen von mehr Fragen richtig zu bestimmen.

Inhalt

1. Einleitung

In Wissensdatenbanken wie Freebase sind enorme Mengen an Allgemeinwissen gespeichert. Alleine Freebase enthält etwa 2.9 Milliarden Fakten in der Form von Triples. Triples sind 3-Tupel der Form (Subjekt, Relation/Prädikat, Objekt). Obwohl die Daten strukturiert sind, kann es durchaus sehr schwierig sein auf einzelne Informationen zuzugreifen, sogar für Experten. Dies ist in erster Linie der enormen Menge an Daten geschuldet. Um auf die Daten in einer Wissensdatenbank zuzugreifen kann man eine strukturierte Abfragesprache wie beispielsweise SPARQL verwenden. In SPARQL werden eine Menge an Triples definiert. Hierbei können auch Variablen verwendet werden. Beim Ausführen der Abfrage werden die definierten Triples mit denen in der Datenbank abgeglichen. Erfolgreiches Formulieren von SPARQL-Abfragen setzt allerdings Wissen über die Struktur der Daten in der Wissensdatenbank voraus. Selbst wenn man die Syntax beherrscht, kann es deshalb trotzdem schwierig sein SPARQL-Abfragen zu formulieren. Das macht die Verwendung von SPARQL für Laien unmöglich. Wenn man beispielsweise die Antwort auf die Frage „Wer ist der CEO von Apple?“ wissen möchte, muss man SPARQL-Abfrage ähnlich der Folgenden auf einer Wissensdatenbank ausführen:

```
select ?name where {
    Managing_Director job_title.people with this title ?0 .
    ?0 employment_tenure.company Apple_Inc .
    ?0 employment_tenure.person ?name
}
```

Beachten Sie hierbei, dass in diesem Fall das Wort „CEO“ in der SPARQL-Abfrage gar nicht vorkommt. Auch kein Wort in den Relationen der SPARQL-Abfrage kommt in der Frage vor. Es wäre natürlich viel einfacher, wenn man seine Frage in natürlicher Sprache stellen könnte. Deshalb ist Question Answering auf Wissensdatenbanken wie Freebase in letzter Zeit Gegenstand mehrerer Veröffentlichungen. Das Question Answering System, welches von dieser Bachelorarbeit betrachtet wird, heißt Aqqu [2] (im Folgenden auch das System genannt) und wurde von Hannah Bast und Elmar Haussmann entwickelt. Aqqu ist in der Lage eine Frage in natürlicher, englischer Sprache in eine SPARQL-Abfrage zu übersetzen. Das funktioniert bei einfachen Fragen ziemlich gut. In einer einfachen werden eine oder maximal zwei Entitäten genannt. Außerdem beinhaltet die Übersetzung

einer einfachen Frage nur eine (möglicherweise k-äre) Relation. Offensichtlich will man jedoch auch manchmal die Antwort auf eine komplexere Frage wissen. Wenn man komplexere Fragen mit mehr genannten Entitäten oder mehr Relationen stellt, ist Aqqu nicht in der Lage diese zu beantworten. Abbildung 1 zeigt was passiert, wenn man Aqqu eine komplexere Frage stellt: Es werden einige Wörter der Frage einfach ignoriert. Statt der komplexen Frage wird eine einfachere Frage beantwortet.

```
2016-06-05 21:55:57,762 : INFO : console_translator : Done translating query: who is wife of the president of america
2016-06-05 21:55:57,762 : INFO : console_translator : #candidates: 150
2016-06-05 21:55:57,763 : INFO : console_translator : SPARQL query: PREFIX fb: <http://rdf.freebase.com/ns/>
 SELECT DISTINCT ?1 where {
 fb:m.09c7w0 fb:government.governmental_jurisdiction.governing_officials ?0 .
 ?0 fb:government.government_position_held.basic_title fb:m.060c4 .
 ?0 fb:government.government_position_held.office_holder ?1 .
 FILTER (?1 != fb:m.060c4 && ?1 != fb:m.09c7w0)
} LIMIT 300
2016-06-05 21:55:57,763 : INFO : console_translator : Result: Benjamin Harrison (m.01b61) James Monroe (m.042d1) Will
iam Henry Harrison (m.0835q) James A. Garfield (m.0b22w) Franklin D. Roosevelt (m.02yy8) George Washington (m.034rd)
George Bush (m.09b6zr) Gerald Ford (m.0c_md_) John Quincy Adams (m.03_nq) Jimmy Carter (m.042kg) William Howard Taft
(m.083pr) John F. Kennedy (m.0d3k14) Abraham Lincoln (m.0gzh) Barack Obama (m.02mjmr) Bill Clinton (m.0157m) James Ma
dison (m.0424m) Ronald Reagan (m.06c0j) Thomas Jefferson (m.07cbs) Harry S. Truman (m.09bg4l) Theodore Roosevelt (m.0
7hyk) James Buchanan (m.042fk) Lyndon B. Johnson (m.0f7fy) John Tyler (m.042dk) James K. Polk (m.042f1) Ulysses S. Gr
ant (m.07t2k) Dwight D. Eisenhower (m.028rk) George H. W. Bush (m.034ls) Richard Nixon (m.06c97) Woodrow Wilson (m.08
3q7) Zachary Taylor (m.08959) Herbert Hoover (m.03kdl) Calvin Coolidge (m.01vbl) William McKinley (m.083p7) John Adam
s (m.03_js) Rutherford B. Hayes (m.06g42) Chester A. Arthur (m.01vb2) Warren G. Harding (m.081t6) Grover Cleveland (m
.038w8) Martin Van Buren (m.04_0m) Franklin Pierce (m.02z51) Andrew Jackson (m.0rlz) Andrew Johnson (m.0rmg) Millard
Fillmore (m.04_13)
```

Abbildung 1: Aqqu versucht die Frage „Wer ist die Ehefrau des Präsidenten der Vereinigten Staaten“ zu beantworten. Da die korrekte Übersetzung mehrere Relationen enthalten würde, ist Aqqu nicht in der Lage die Frage zu beantworten. Stattdessen wird die Antwort auf die Frage „Wer ist der Präsident der Vereinigten Staaten?“ geliefert.

Des Weiteren hat Aqqu Probleme mit Superlativ-Fragen und Temporal-Fragen: Aqqu behandelt Superlative wie normale Adjektive und ist dementsprechend nicht in der Lage Fragen zu beantworten, die nach Superlativen fragen. Auch wenn die Fragen eine Zeitspanne oder irgendeine andere Form von zeitlicher Beschränkung enthält, wird die Übersetzung von Aqqu diese meistens ignorieren. In dieser Bachelorarbeit werden im Folgenden mehrere Erweiterungen vorgestellt, mit denen Aqqu in der Lage ist, solche Fragen zu beantworten. Diese Erweiterungen werden nach dem Übersetzungsvorgang von Aqqu ausgeführt und verwenden die beim Übersetzungsvorgang berechneten Daten.

Eine andere Verbesserung, welche in dieser Arbeit vorgestellt wird, ist ein verbessertes Answer Type Matching (dt. Antworttyperkennung). Im ursprünglichen Aqqu-System wurde dafür lediglich das Fragewort betrachtet um zu bestimmen ob der Antworttyp einer möglichen Übersetzung korrekt ist. Bei einer Frage, welche beispielsweise mit „Wer“ beginnt, wird als Antwort eine Person, ein Charakter oder eine Organisation erwartet. Dies ist zwar ziemlich effektiv, ein ausgefeilterer Mechanismus könnte allerdings die Genauigkeit des System möglicherweise

signifikant verbessern. So erwartet man beispielsweise als Antwort auf die Frage „Wer ist Johnny Depp?“ eher die Liste der Berufe des Prominenten als seinen Namen. Answer Type Matching ist besonders wichtig, wenn die Wörter in der Frage nur sehr schlecht den Relationen und Entitäten der Wissensdatenbank zugeordnet werden können. Das ist vor allem dann der Fall, wenn das Verb der Frage „sein“ ist. Das ist sehr oft der Fall. Um dieses Problem in den Griff zu bekommen, verwende ich einen auf Machine Learning basierenden Klassifizierer. Dieser berücksichtigt eine große Menge an Features (dt. Merkmale). Er bestimmt dann den Antworttyp zu einer gegebenen Frage. Dabei werden die Entitätstypen von Freebase verwendet.

1.1 Beiträge

Ich betrachte das Folgende als meine Beiträge:

- Ein Mechanismus, durch den es Aqqu möglich ist komplexere Fragen wie Superlativ-Fragen, Temporal-Fragen oder Fragen, dessen Übersetzung mehrere Relationen enthält, in SPARQL zu übersetzen.
- Einen auf Machine Learning basierenden Mechanismus um den Antworttyp einer Frage mit höherer Genauigkeit zu bestimmen. Dieser verwendet die Typen von Freebase und berücksichtigt eine Reihe verschiedener Features.

1.2 Grundlagen von Natural Language Processing

In diesem Abschnitt werden einige grundlegende Konzepte von Natural Language Processing (NLP) erläutert, die im Folgenden häufiger erwähnt werden.

1.2.1 Part-of-Speech Tagging

Beim Part-of-Speech Tagging, kurz POS-Tagging, werden den Wörtern in einem Satz POS-Tags zugeordnet. Jedes POS-Tag repräsentiert hierbei eine Wortart. Mögliche POS-Tags wären NN für Nomen, NNP für Eigennamen, JJ für Adjektive und VB für ein Verb in Grundform. Es gibt noch einige weitere POS-Tags.

1.2.2 Dependency Tree

Aus dem Dependency Tree eines Satzes kann man die Abhängigkeiten zwischen den einzelnen Wörtern ablesen. Siehe Abbildung 2 als Beispiel.

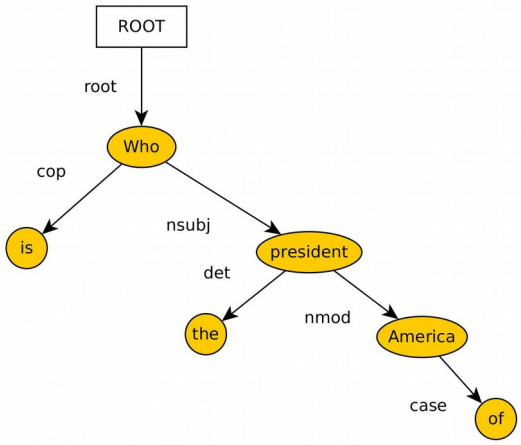

Abbildung 2: Beispiel eines Dependency Tree für die Frage „Wer ist der Präsident der Vereinigten Staaten?“

1.2.3 WordNet

WordNet [7] ist eine große lexikalische Datenbank für die englische Sprache. Die Wörter sind in Mengen von Synonymem, genannt Synsets, organisiert. Des Weiteren kann man mittels WordNet unter anderem die Hyponyme (Unterbegriffe) und Hyperonyme (Überbegriffe) zu einem Wort erhalten.

1.3 Ursprüngliches System

Das System, welches von dieser Bachelorarbeit erweitert werden soll, heißt Aqqu. Es ist in der Lage einfache Fragen in SPARQL-Abfragen zu übersetzen. Diese ermitteln, ausgeführt auf einer Wissensdatenbank, die Antwort auf die gestellte Frage. Aqqu erkennt die Entitäten der Wissensdatenbank in der Frage. Die restlichen Wörter in der Frage werden dann mit den Relationen der Wissensdatenbank abgeglichen. Wegen der hohen Variabilität natürlicher Sprache ist diese

Aufgabe sehr schwierig. Zum Beispiel können mehrere Entitäten in der Wissensdatenbank denselben Namen haben. Des Weiteren kann es auch mehrere Namen für ein und dieselbe Entität geben.
Aqqu arbeitet in mehreren Phasen. Mehr Details sind in der entsprechenden Veröffentlichung [2] zu finden.

1.3.1 Identifizierung der Entitäten

In der ersten Phase werden alle Entitäten, welche in der Frage genannt werden, identifiziert. Da es in diesem Schritt noch sehr häufig zu Mehrdeutigkeiten kommt, ist es nicht möglich die Entitäten mit Sicherheit zu bestimmen. Deshalb ist das Ergebnis dieser Phase eine Menge möglicher Entitäten. In den folgenden Phasen werden diese Mehrdeutigkeiten aufgelöst.

Diese Phase beginnt mit der Bestimmung der POS-Tags der Wörter in der Frage. Dafür wird der Stanford POS-Tagger [12] verwendet. Mit diesen POS-Tags werden alle möglichen Wortsequenzen einer gegebenen Frage bestimmt, welche eine Erwähnung einer Entität darstellen könnten. Das heißt es werden alle Entitäten der Wissensdatenbank bestimmt, dessen Name oder Alias einer dieser Wortsequenzen entspricht. Für Aliasse wird der CrossWiki [15] Datensatz verwendet. Damit war es möglich etwa 44 Millionen Entitäten mit etwa 60 Millionen Aliassen zu identifizieren. In einem letzten Schritt werden Scores (dt. Wertungen) für die identifizierten Entitäten berechnet. Aqqu verwendet die Anzahl der Vorkommnisse einer Entität im ClueWeb12 [6] Datensatz als Popularitätswertung. Die Wahrscheinlichkeit, dass eine Entität von einem gewissen Alias gemeint ist, ist ein anderer Score. Dieser wird teils von CrossWiki zur Verfügung gestellt und muss teils über die Popularitätswertung berechnet werden.

1.3.2 Generierung von Kandidaten und Template Matching

In der zweiten Phase wird eine Menge von Templates verwendet um mögliche Übersetzungen, im Folgenden Kandidaten genannt, zu generieren. Abbildung 3 zeigt die Menge der Templates. Das erste Template besteht lediglich aus einem Platzhalter für eine Entität, einem Platzhalter für eine binäre Relation, sowie einem Antwortknoten. Die beiden anderen Templates sind komplexer und enthalten Platzhalter für eine k-äre (k>2) Relation und teilweise mehrere Platzhalter für Entitäten. k-äre Relationen werden in Wissensdatenbanken meist mittels einem Zwischenobjekt realisiert. Platzhalter für diese Zwischenobjekte sind in den beiden komplexeren Templates enthalten.

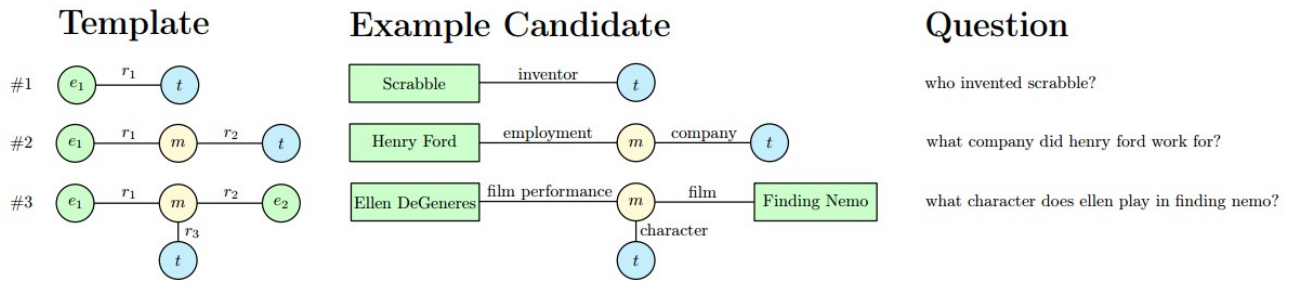

Abbildung 3: Templates und Beispielkandidaten mit den zugehörigen Fragen. Ein Template kann aus Platzhaltern für Entitäten e, Relationen r, Zwischenobjekten m und dem Antwortknoten t bestehen.

1.3.3 Relation Matching

In dieser Phase werden die Wörter in der Frage mit den Relationen in den generierten Kandidaten abgeglichen und ihnen zugeordnet. Dabei wird berechnet wie gut die Relationen zu den Wörtern in der Frage passen. Dafür werden die lemmatisierten[1] Wörter der Relationen im Kandidaten und die lemmatisierten Wörter in der Frage betrachtet. All jene Wörter, die bereits zu einer Entität zugeordnet wurden, werden hierbei ignoriert. Es sind vier verschiedene Arten von Zuordnungen möglich: Bei Literal Matches stimmen Wörter wörtlich überein, bei Derivation Matches stammen Wörter voneinander ab, bei Synonym Matches werden Synonyme berücksichtigt. Für Context Matches wird für jede Relation eine Menge von Indikator-Wörtern berechnet. Indikator-Wörter werden in einem Text verwendet um eine Relation auszudrücken. Diese werden dann für Context Matches berücksichtigt. Für jeden dieser Zuordnungstypen wird außerdem ein Score berechnet. Mehrere Wörter können durch mehrere Zuordnungen verschiedener Typen einer Relation zugeordnet werden.

1.3.4 Answer Type Matching

Es wird lediglich überprüft ob der vom Fragewort implizierte Antworttyp mit dem Antworttyp des Kandidaten übereinstimmt. Wenn beispielsweise eine Frage mit „Wer“ beginnt, dann erwartet man ein Ergebnis mit Typ Person, Charakter oder Organisation. Diese Bachelorarbeit wird unter anderem das Answer Type Matching verbessern.

1.3.5 Features und Ranking

Für das Ranking der Kandidaten werden 23 verschiedene Features für jeden Kandidaten berechnet. Die meisten werden aus den Scores, welche oben erwähnt wurden, hergeleitet. Ein weiteres

1 Das Lemma eines Wortes bezeichnet die Grundform dieses Wortes

wichtiges Feature ist das sogenannte N-Gram Relation Matching Feature. Dadurch wird der Zusammenhang zwischen n-grams der Frage und Relationen der Kandidaten berücksichtigt. Mit diesen Features werden die Kandidaten geordnet. Der beste Kandidat wird als Ergebnis der Übersetzung zurück gegeben. Es wurden zwei auf Supervised Learning basierende Ranking-Methoden untersucht. In Pointwise Ranking werden die Kandidaten über einen Score geordnet. Dieser wurde zuvor mittels eines Klassifizierers gelernt. In Pairwise Ranking wird ein ebenfalls ein Klassifizierer gelernt. Dieser bestimmt für ein gegebenes Paar von Kandidaten, welcher höher eingeordnet werden soll.

1.3.6 Pruning

Auf manche Fragen gibt es keine Antwort. Es sollte also einen Mechanismus geben, welcher alle Kandidaten verwirft, die offensichtlich nicht die richtige Übersetzung darstellen. Dabei wird außerdem der Ranking Prozess beschleunigt, da beim Ranking weniger Kandidaten berücksichtigt werden müssen. Hierbei gibt es zwei Varianten: Bei *Hard-Pruning* werden alle Kandidaten des falschen Antworttyps oder mit zu vielen Nullen in bestimmten Features verworfen. Mit einem *Pruning Klassifizierer* wird ein zuvor gelernter Klassifizierer verwendet um schlechte Kandidaten zu verwerfen.

1.3.7 Evaluierung

Das fertige System wurde auf den beiden Datensätzen Free917 [5] und WebQuestions [3] ausgewertet und mit anderen Systemen verglichen. Die verwendeten Evaluierungsmaße wurden dabei auf die Datensätze angepasst. Dabei werden die anderen Systeme um ca. 10 % übertroffen.

1.4 Herausforderung und Herangehensweise

1.4.1 Beantworten von komplexen Fragen

Wie in Abschnitt 1.2 gezeigt wurde, ist das grundsätzliche Vorgehen von Aqqu zunächst einmal Kandidaten zu generieren. Diese werden dann beim Ranking geordnet und der beste Kandidat wird zurückgegeben. Um komplexere Fragen beantworten zu können, müssen offensichtlich weitere Kandidaten generiert werden. Man könnte weitere Templates hinzuzufügen um weitere Fragetypen beantworten zu können. Da natürliche Sprache sehr variabel ist, wäre die Anzahl der neuen Kandidaten dementsprechend hoch. Auch die Anzahl der generierten Kandidaten würde damit

vermutlich stark steigen. Durch das generieren und sortieren der vielen Kandidaten würde die Laufzeit ebenfalls ansteigen.

Es muss also ein Weg gefunden werden Kandidaten für eine große Menge unterschiedlicher, komplexer Fragen zu generieren. Dabei darf die Menge der generierten Kandidaten nicht zu groß werden. Um diese Herausforderung zu lösen habe ich mich entschieden das grundsätzliche Vorgehen von Aqqu nicht zu ändern und auch keine neuen Templates hinzuzufügen. Stattdessen wird zunächst das ursprüngliche System ausgeführt und eine Menge von Kandidaten wird generiert. In einer neuen Phase danach werden die generierten Kandidaten verwendet um neue Kandidaten zu generieren. Das hat auch den Vorteil, dass die neuen Kandidaten nicht komplett neu generiert werden müssen. Diese neuen Kandidaten sind Erweiterungen der alten Kandidaten, bei denen die alten Kandidaten durch eine Relation o.ä. erweitert wurden. Die Hoffnung ist, dass dabei auch die richtigen Kandidaten für komplexe Fragen generiert werden können ohne die Anzahl der generierten Kandidaten zu stark zu erhöhen.

1.4.2 Answer Type Matching

Die Herausforderung beim Answer Type Matching ist unter anderem eine ausreichende Menge an Features aus einer Frage zu berechnen, die es dann einem Klassifizierer möglich macht den richtigen Antworttyp einer Frage zu bestimmen. Dafür habe ich mich von [10] inspirieren lassen. Ich verwende verschiedene syntaktische und semantische Features, welche es oft ermöglichen den richtigen Antworttyp zu bestimmen.

2. Verwandte Arbeiten

Viele Question Answering Systeme, welche die Antwort einer gestellten Frage aus einer Wissensdatenbank extrahieren und in der Lage sind komplexe Fragen zu beantworten, verwenden QALD [14] zur Evaluation. QALD (Question Answering over Linked Data) ist eine Serie an Evaluierungskampagnen, die in 2011 startete. Momentan gibt es sechs Benchmarks, eine für jedes Jahr. Jede Benchmark hat hierbei einen anderen Fokus. Die komplexen Fragen, welche auch diese Bachelorarbeit behandelt, sind in QALD häufiger anzutreffen als in Free917 [5] und WebQuestions [3]. Aqqu [2] wurde bisher nicht auf den QALD Benchmarks evaluiert. Dies ist hauptsächlich deshalb der Fall, weil die Anzahl der Fragen in den Trainingssets zu klein ist. Aqqu basiert auf

Supervised Learning und somit ist ein großes Trainingsset essenziell für die Qualität der Antworten von Aqqu. Die meisten an QALD teilnehmenden Systeme basieren deshalb nicht auf Supervised Learning. Des Weiteren ist die größte verwendete Wissensdatenbank DB-pedia. Diese ist um einiges kleiner als Freebase (4 Millionen vs. 40 Millionen Entitäten).

Im Folgenden stelle ich einige Arbeiten vor, die in der Lage sind die komplexen Fragen, welche diese Bachelorarbeit behandelt, zu beantworten. Da diese Arbeiten mit QALD evaluiert wurden, unterscheidet sich jedoch die Herausforderung der Arbeiten aus den eben genannten Gründen von den Herausforderungen dieser Bachelorarbeit.

- In [18] wird zunächst eine Graphrepräsentation der Frage aus der Frage und deren Dependency Tree generiert. Diese kann dabei auch durchaus mehrere Relationen enthalten. Hierfür wird ein sogenanntes Paraphrase Dictionary vorberechnet, das Phrasen der Frage auf Relationen abbildet. In einem nächsten Schritt werden die Entitäten identifiziert und die Graphrepräsentation der Frage wird mit dem Graph der Wissensdatenbank abgeglichen. Es wird der Subgraph im Graph der Wissensdatenbank gesucht, der am besten auf die Graph Repräsentation der Frage passt. Dafür wird ein Score für jeden Kandidaten mittels einer Formel berechnet. Aus diesem kann dann sowohl die Antwort der Frage, als auch die Übersetzung in SPARQL entnommen werden.
- In [8] wird die Frage mit dem Dependency Tree, den identifizierten Entitäten der Frage und mithilfe diverser Systeme, wie beispielsweise WordNet [7], zu Ontology Triples umgewandelt. Hier kann es zu Mehrdeutigkeiten kommen. In einer SPARQL Generation Phase werden dann aus den Ontology Triples mehrere SPARQL-Abfragen generiert. Ein Score wird für jeden Kandidaten berechnet und die SPARQL-Abfrage mit dem höchsten Score wird zurückgegeben.
- In [16] wird das Übersetzen einer natürlichsprachlichen Frage in SPARQL als Auflösen von mehreren Mehrdeutigkeiten betrachtet. Zunächst werden die Phrasen der Frage erkannt und Relationen oder Entitäten zugeordnet. Dann werden Triples generiert, die jedoch die Mehrdeutigkeiten im vorherigen Schritt berücksichtigen. Schließlich werden alle Mehrdeutigkeiten auf einmal mittels eines Integer Linear Program aufgelöst. Danach kann die SPARQL-Abfrage generiert und die Antwort aus der Wissensdatenbank extrahiert werden.

Die folgenden Arbeiten beschäftigen sich mit Fragen, die eine zeitliche Bedingung beinhalten.

- In [17] verwenden die Autoren eine Wissensdatenbank mit n-Tuple Assertions als Einträge. Diese bestehen aus einem Subjekt, einem Prädikat und mehreren Objekten. Sie sind semantisch reichhaltiger als die Triples von Freebase. Mit einem neuen Mechanismus zur Antwortextraktion können damit komplexe Fragen, wie solche, die zeitliche Bedingungen enthalten, beantwortet werden.
- In [13] wird ein System zur Beantwortung von Temporal-Fragen vorgestellt, dass ein beliebiges, bereits existierendes Question Answering System erweitern soll. Wird eine Frage gestellt, dann wird diese analysiert. Falls sie eine zeitliche Beschränkung enthält wird sie in mehrere einfache Fragen aufgeteilt. Die einfachen Fragen werden dann an das Question Answering System weitergeleitet und von diesem beantwortet. Die Antworten werden danach interpretiert und zusammengefügt. Die richtige Antwort auf die originale Frage wird zurückgegeben. Hier wird nur die Antwort zurückgegeben und keine Übersetzung in SPARQL.

Da die Erkennung des Antworttyps ein wichtiger Bestandteil vieler Question Answering Systeme ist, gibt es dazu einige Veröffentlichungen. Diese unterscheiden sich unter anderem dadurch von dem Answer Type Matching System dieser Bachelorarbeit, dass die verwendeten Antworttypen selbst bestimmt wurden und häufig hierarchisch aufgebaut sind. Da Freebase selbst Entitätstypen mitbringt, habe ich mich entschieden diese zu verwenden.

- In [10] wird ein hierarchischer Klassifizierer verwendet. Dieser besteht aus einem groben Klassifizierer, der zunächst die Klasse des Antworttyps aus 6 Möglichkeiten berechnet. Ein feiner Klassifizierer bestimmt dann den Antworttyp. Insgesamt werden 50 verschiedene Antworttypen verwendet. Zu 85% ist der identifizierte Antworttyp korrekt.
- In [9] haben die Autoren erkannt, dass ein Mensch den Antworttyp einer Frage an nur sehr wenigen, kurzen Textpassagen erkennen kann. Um diese zu ermitteln, wurden mehrere Methoden getestet. Unter anderem die Erkennung durch sogenannte Conditional Random Fields und Zustandsübergangsmodelle.

3. Erweiterungen in der Generierung von Kandidaten

Wenn man komplexe Fragen wie zum Beispiel „Wer ist die Ehefrau des Präsidenten der Vereinigten Staaten?“ stellt, ist Aqqu [2] nicht in der Lage diese zu beantworten. Das ist deshalb der Fall, weil kein Template, wie sie in Abschnitt 1.3 vorgestellt wurden, auf die Frage passt. In Abbildung 4 kann man sehen, dass der Kandidat, welcher die Übersetzung für die Frage darstellt, zwei Zwischenknoten und damit 2 k-äre Relationen enthält. Die Templates enthalten allerdings nur maximal einen solchen Zwischenknoten. Bei solchen Fragen ist Aqqu meistens nicht in der Lage alle Wörter der Frage einer Entität oder Relation in einem Kandidaten zuzuordnen. Es gibt also für jeden generierten Kandidaten eine Menge an Wörtern, welche noch weiteren Relationen oder Entitäten zugeordnet werden können.

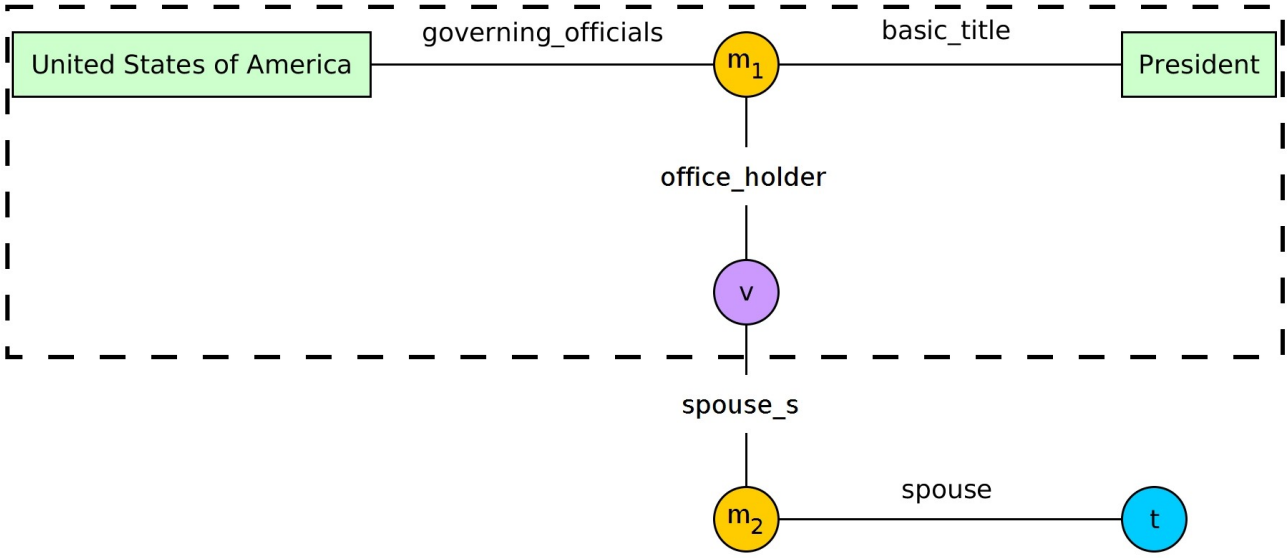

Abbildung 4: Kandidat der komplexen Frage „Wer ist die Ehefrau des Präsidenten der Vereinigten Staaten?“. Der Teil des Kandidaten in der gestrichelten Linie ist die Übersetzung für die Frage „Wer ist der Präsident der Vereinigten Staaten?“, wenn v der Antwortknoten ist.

Sei q eine komplexe Frage, welche nicht von Aqqu beantwortet werden kann. Sei c(q) der Kandidat, welcher die korrekte Übersetzung von q darstellt. Sei A(q) die Menge der Kandidaten, welche von Aqqu berechnet werden. Da q nicht von Aqqu beantwortet werden kann, ist c(q) kein Element von

A(q). Betrachte die Frage q="Wer ist die Ehefrau des Präsidenten der Vereinigten Staaten?". Der Kandidat c', welcher im Ranking ganz oben steht, beantwortet die Frage „Wer ist der Präsident der Vereinigten Staaten?". Das Wort „Ehefrau" wurde also von diesem Kandidaten ignoriert. In der gestrichelten Box von Abbildung 4 kann man c' sehen. Um c' zu c(q) zu erweitern muss man an den Antwortknoten von c' die spouse-Relation (spouse: dt. Ehepartner) und einen neuen Antwortknoten anhängen. Da das Wort „Ehefrau" der spouse-Relation zugeordnet werden kann, kann jedes Wort der Frage zu einer Entität oder Relation in c(q) zugeordnet werden. Bei der Zuordnung von Wörtern an Relationen wird das in Abschnitt 1.3.3 erläuterte Relation Matching verwendet.

Manchmal kann es vorkommen, dass ein Wort vom ursprünglichen Aqqu System mittels eines Context Matches einer Relation zugeordnet wird, zu der es eigentlich nicht zugeordnet werden sollte. Der Score dieser Zuordnung ist dabei dementsprechend gering. Solche Wörter sollten bei einer Erweiterung neuen Relationen zugeordnet werden können. Deshalb werden alle Wörter, die mit einem Context Match einer Relation zugeordnet wurden und der Score des Matches kleiner ist als ein gewisser Grenzwert (0.05), als noch nicht zugeordnet betrachtet. Kann das Wort einer neuen Relation zugeordnet werden, so wird die alte Zuordnung entfernt.

3.1 Kandidaten zur Erweiterung bestimmen

Das ursprüngliche Aqqu-System generiert eine große Menge an Kandidaten. Würden alle diese Kandidaten erweitert werden, dann würden für jeden Kandidaten in dieser Menge weitere Kandidaten generiert werden. Die resultierende Menge an Kandidaten wäre viel zu groß, sodass das Ranking nicht in angemessener Zeit berechnet werden kann. Es muss also eine kleinere Auswahl an Kandidaten getroffen werden, die dann im Folgenden erweitert wird. Hier habe ich die folgende Methode ausprobiert. Weitere Möglichkeiten werden in Abschnitt 7 diskutiert.

Top k des Rankings: Im Allgemeinen sieht man durch Beobachtung, dass Kandidaten, welche erweitert werden können um komplexe Fragen zu beantworten, im Ranking ziemlich weit oben stehen. Eine Idee wäre also, die obersten k Kandidaten im Ranking in der Erweiterung zu berücksichtigen. Je höher der Wert von k ist, desto mehr Kandidaten werden generiert. Es steigt also sowohl Laufzeit als auch Qualität der Übersetzung. Momentan werden die 10 besten Kandidaten in der Erweiterung berücksichtigt, was ein guter Kompromiss zu sein scheint. Ein Nachteil dieser Methode ist, dass man das Ranking berechnen muss, um die k besten Kandidaten zu berechnen. Dies kostet Laufzeit.

3.2 Knoten zur Erweiterung bestimmen

Es gibt mehrere Gründe um einen Kandidaten zu erweitern. Zum einen, wie in den Beispielen oben, um komplexe Fragen, deren Übersetzungen mehrere Relationen beinhalten, zu beantworten. Aber auch für Temporal-Fragen oder Superlativ-Fragen müssen Kandidaten erweitert werden. Auf diese wird in Abschnitt 3.4 bzw. 3.5 näher eingegangen. Je nach Grund für die Erweiterung, ist eine Erweiterung an vier Knoten sinnvoll. Sei w das Wort, welches noch nicht zugeordnet werden konnte:

(1) **Eine Erweiterung am Antwortknoten**, wie im Beispiel oben.

(2) **Eine Erweiterung an dem Knoten auf den sich das w laut dem Dependency Tree der Frage bezieht**. Dieser Knoten x wird wie folgt berechnet: Zunächst wird der Dependency Tree der Frage mittels dem Stanford Parser [12] berechnet. Aus einem Dependency Tree kann man die Abhängigkeiten zwischen den Wörtern ablesen, wie Abbildung 2 in Abschnitt 1.2 zeigt. Starte nun an dem Wort, welches noch nicht zugeordnet werden konnte und laufe im Dependency Tree in Richtung Wurzel. Sobald man im Dependency Tree ein Wort erreicht hat, welches bereits einer Entität zugeordnet werden konnte, so ist x der Knoten dieser Entität im Kandidaten.

(3) **Eine Erweiterung am Zwischenknoten**. Oft ist eine Erweiterung am Zwischenknoten notwendig. Ein Beispiel hierfür wäre die Temporal-Frage „Wer war der erste Präsident der Vereinigten Staaten“. Sie fragt nach der Person, die als erstes das Amt des Präsidenten der USA übernommen hat. Der beste Kandidat des ursprünglichen Aqqu-Systems ignoriert hierbei das Wort „erste“. Das Datum der Übernahme des Präsidentenamtes ist in Freebase aber keine Eigenschaft der jeweiligen Personen, sondern der „hat Titel als Regierungsbeamter“-Relation. Somit muss eine Erweiterung am Zwischenknoten der k-ären Relation stattfinden. Es ist aufgrund von vorherigen Erweiterungen möglich, dass ein Kandidat mehrere Zwischenknoten hat. Dann muss mit Hilfe des Dependency Tree herausgefunden werden welcher der richtige Zwischenknoten ist. Dazu wird der Knoten der Entität oder Relation im Kandidaten berechnet, auf den sich das noch nicht zugeordnete Wort bezieht, berechnet. Der gesuchte Zwischenknoten ist derjenige, der am nächsten an dem berechneten Entitätsknoten oder der berechneten Relation im Kandidaten liegt.

(4) **Ein ehemaliger Antwortknoten**, der aufgrund einer vorherigen Erweiterung jedoch nur noch eine Variable ist. Ein Beispiel hierfür wäre „Wer ist die Ehefrau des größten Präsidenten der Vereinigten Staaten?“. Um das Wort „größte“ zuzuordnen muss der Kandidat für die Frage „Wer war Ehefrau eines Präsidenten der Vereinigten Staaten ?“ an der Variable für den Präsidenten mit

der Körpergrößen-Relation erweitert werden. Diese Variable war zuvor Antwortknoten für den Kandidaten, der die Frage „Wer war Präsident der Vereinigten Staaten?“ beantwortet.

Die folgende Methode wurde verwendet um die Knoten zu bestimmen. Weitere Methoden werden in Abschnitt 7 diskutiert.

Bestimmung der Knoten vor der Ausführung: Die Knoten, an denen eine Erweiterung sinnvoll ist, wurden vor der Ausführung bestimmt. Je nach Grund der Erweiterung eine Untermenge der oben genannten vier Knoten. Es führt dazu, dass die Menge der neu generierten Kandidaten relativ hoch sein kann, was auch die Laufzeit ansteigen lässt. In den folgenden Kapiteln wird stets erwähnt, welche Knoten für das Erweitern im jeweiligen Fall sinnvoll ist.

3.3 Zwei Fragen in einer

Dieser Abschnitt behandelt Fragen die eine Komposition aus zwei einfachen Fragen ist. Das Beispiel „Wer ist die Ehefrau des Präsidenten der Vereinigten Staaten?“ ist von diesem Typ. Diese Frage kann in zwei einfache Fragen aufgespalten werden: „Wer ist der Präsident der Vereinigten Staaten?“ und „Wer ist die Ehefrau von Barack Obama?“, wobei „Barack Obama“ die Antwort auf die erste Frage ist. In diesem Fall muss also die erste einfache Frage zuerst beantwortet werden. Die Antwort der ersten Frage kann dann verwendet werden um die zweite Frage zu stellen. Die Antwort der zweiten Frage ist dann die Antwort auf die originale Frage. Wie man diese Fragen beantworten kann wurde in den vorherigen Abschnitten schon angedeutet. Im Folgenden werden diese Art von Fragen „Fragen von Typ 1“ genannt.

Im Gegensatz dazu stehen die „Fragen von Typ 2“. Ein Beispiel hierfür wäre „Welche Schauspieler haben einen Oscar gewonnen?“. Auch diese Frage kann man in 2 einfache Fragen aufteilen: „Wer ist ein Schauspieler?“ und „Wer hat einen Oscar gewonnen?“. Diesmal ist das Ergebnis der originalen Frage allerdings der Schnitt der Antwortmengen der beiden einfachen Fragen. Das sind all jene Personen, die sowohl Schauspieler sind als auch einen Oscar gewonnen haben. Man könnte die zweite Frage auch als Bedingung auffassen, welche die Elemente in der Antwortmenge der ersten Frage erfüllen müssen.

Betrachte nun einen Kandidaten c, der erweitert werden soll, um die Fragen dieser Typen beantworten zu können. Sei unmatched(c) die Menge aller Worte, die noch keiner Relation und

noch keiner Entität in dem Kandidaten zugeordnet werden konnten. Falls diese Menge leer ist, wird der zugehörige Kandidat nicht erweitert, da er schon die Frage beantworten könnte.

Es wird, wie in Abschnitt 3.2 erklärt, die Menge der Knoten bestimmt, an denen eine Erweiterung sinnvoll ist. Sinnvoll ist hier eine Erweiterung am Antwortknoten.

Um einen Kandidaten c an einem gegebenen Knoten v so zu erweitern, dass er eine Frage dieser Typen beantworten kann, gehe ich wie folgt vor: Mit einer einzigen SPARQL-Abfrage kann man alle möglichen Erweiterungen erhalten. Dies ist eine Menge an möglicherweise k-ären Relationen, die v als Subjekt haben. Im nächsten Schritt werden diese Relationen und die Wörter in unmatched(c) einander zugeordnet und der Kandidat wird um die Relationen erweitert. Wie die Zuordnung (Relation Matching) funktioniert wurde in Abschnitt 1.2 erläutert. Für jede gefundene Relation, denen mindestens ein Wort zugeordnet werden kann, gibt es also einen neuen Kandidaten. Schließlich wird jedem dieser Kandidaten ein neuer Antwortknoten hinzugefügt.

Bisher wurden nur Kandidaten für Fragen von Typ 1 generiert. Falls eine Frage von Typ 2 vorliegt, so ist ein Kandidat c unter den erweiterten Kandidaten, für den das Folgende gilt: Es wurde zu Beginn eine Entität identifiziert, die jedoch keinen Knoten in dem Kandidaten darstellt. Alle Wörter, die der Entität zugeordnet werden konnten, sind Elemente von unmatched(c). Zudem ist diese Entität in der Antwortmenge des Kandidaten. Siehe hierzu Abbildung 5 als Beispiel. Der Kandidat, welcher die Frage „Wer ist ein Schauspieler?“ beantwortet wurde am Antwortknoten mit der „Auszeichnung gewonnen“-Relation erweitert. Führt man die SPARQL-Abfrage zu diesem Kandidaten aus, dann würden alle Auszeichnungen, die jemals von einem Schauspieler gewonnen wurden, zurückgegeben. Unter anderem wäre auch die Entität „Academy Awards“ in der Ergebnismenge. Diese Entität wurde zu Beginn in der Frage identifiziert, da das Wort Oscar in der Frage vorkommt. Um die originale Frage zu beantworten muss nun der Antwortknoten durch die Entität „Academy Awards“ ersetzt werden und der Antwortknoten ist wieder derselbe, wie vor der Erweiterung.

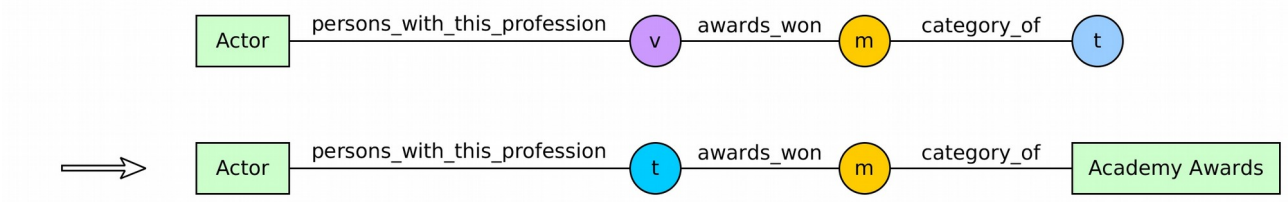

Abbildung 5: Erweiterung eines Kandidaten für die Frage „Welche Schauspieler haben einen Oscar gewonnen?“

3.4 Temporal-Fragen

Temporal-Fragen bezeichnen all jene Fragen, die eine Form von zeitlicher Beschränkung beinhalten. Hierbei gibt es eine Menge an Signalwörtern, die andeuten das es sich bei einer Frage um eine Temporal-Frage handeln könnte. Tabelle 1 zeigt diese (englischen) Wörter.

Fragen nach dem letzten Ereignis in einer Serie von Ereignissen	last, current, currently, at the moment, final, now, newest, latest
Fragen nach dem frühesten Ereignis	First, oldest, earliest
Explizite zeitliche Beschränkung	Q after A, Q following A, Q before A, Q during A Q from A to B, Q between A and B, Q about A - B Q while A, Q for A, Q since A

Tabelle 1: Unterstützte Signalwörter.

3.4.1 Einfache Temporal-Fragen

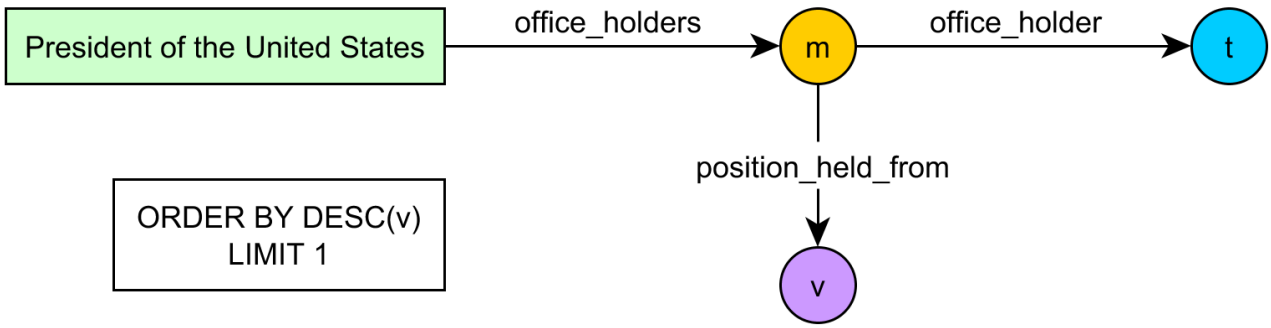

Abbildung 6: Beispiel für einen Kandidaten für eine Frage mit Signalwort aus der zweiten Zeile. Sie lautet: „Wer ist der aktuelle (current) Präsident der Vereinigten Staaten?“

Für die Signalwörter der ersten beiden Zeilen funktioniert die Erweiterung eines Kandidaten c wie folgt: Es wird zunächst, wie in Abschnitt 3.2 beschrieben, die Knoten bestimmt, bei denen eine Erweiterung sinnvoll ist. In diesem Fall ist das der Zwischenknoten und der Antwortknoten. Es werden nun alle möglichen Relationen gesucht, mit denen der Kandidat erweitert werden kann und die als Objekt ein Datum erwarten. Die Wörter in unmatched(c) werden danach den Relationen zugeordnet, falls möglich. Für jede Relation wird nun ein neuer Kandidat generiert. Oft können die Signalwörter nicht den Relationen zugeordnet werden, obwohl dies sinnvoll wäre. Ist das der Fall wird ein Context Match mit Score von mindestens 0.1 hinzugefügt. Der Antwortknoten ändert sich hierbei nicht. Nun wird der SPARQL-Abfrage des Kandidaten eine ORDER BY Klausel hinzu

gefügt. Die Ergebnisse sollen je nachdem, ob das früheste oder das letzte Element gesucht ist, auf- oder absteigend nach dem Objekt der neu hinzugefügten Relation sortiert werden. Außerdem wird das LIMIT der SPARQL-Abfrage auf 1 gesetzt, sodass nur das früheste oder späteste Ergebnis ausgegeben wird. Siehe Abbildung 6 als Beispiel.

3.4.2 Temporal-Fragen mit expliziter zeitlicher Beschränkung

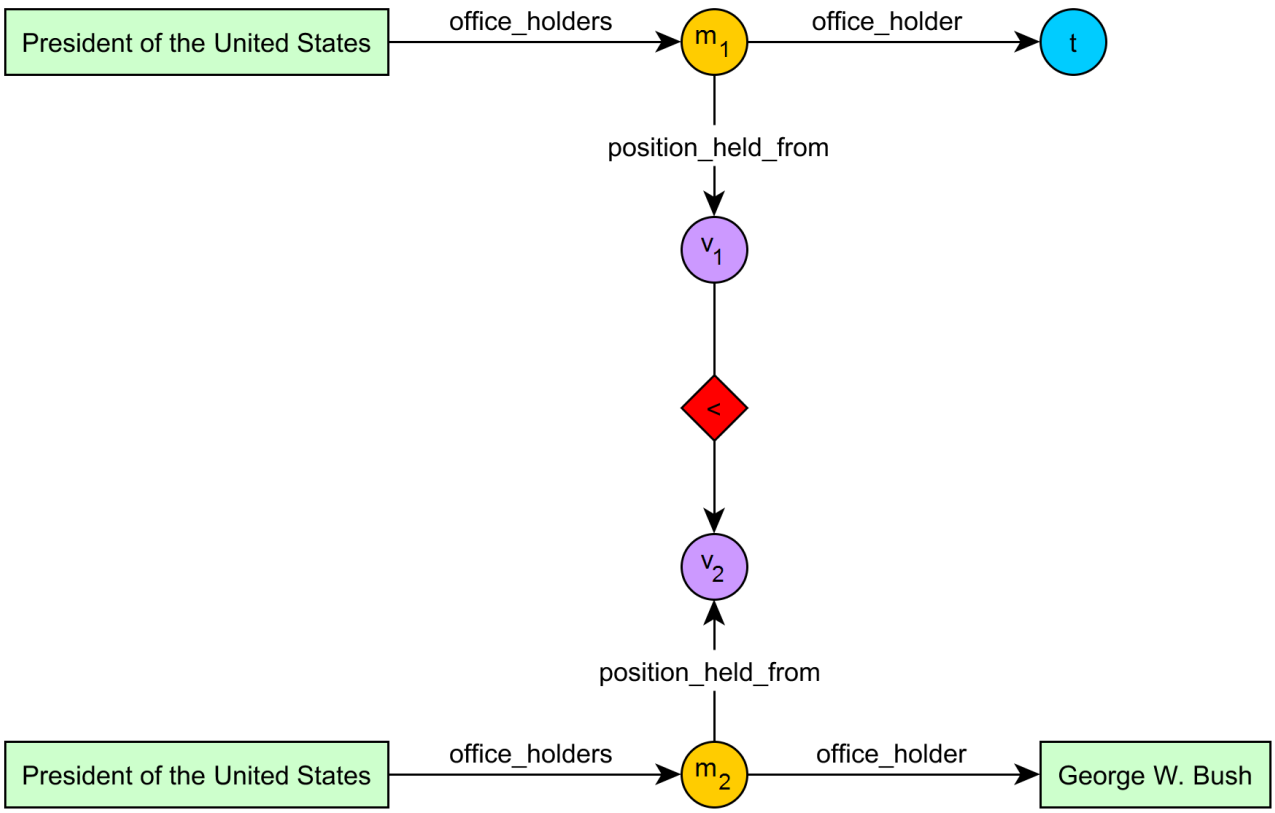

Abbildung 7: Beispiel für einen Kandidaten für eine Frage mit Signalwort aus der letzten Zeile. Diese lautet: „Wer war Präsident der Vereinigten Staaten vor (before) George W. Bush?“. Die rote Raute stellt den Filter dar.

Für die Signalwörter der letzten Zeile funktioniert die Erweiterung eines Kandidaten anders. Zunächst muss die Frage am Signalwort in 2 bzw. 3 Teile aufgeteilt werden. Die Teile sind in Tabelle 1 durch die großen Buchstaben Q, A, B dargestellt. Berechne und erweitere nun Kandidaten zur Teilfrage Q. Dieses Vorgehen ist von [13] inspiriert. Betrachte für einen Kandidaten nun nacheinander die Teile A und B, falls B existiert.

Es gibt drei Fälle. (1) Im Teil A bzw. B wird eine Entität erwähnt, welche auch in der Lösungsmenge des Kandidaten vorkommt. In diesem Fall wird der Kandidat um eine Relation erweitert, welche ein Datum als Objekt erwartet. Nun wird der Kandidat kopiert und der

Antwortknoten der Kopie wird durch die die erwähnte Entität ersetzt. Schließlich wird noch ein Filter[2] hinzugefügt, sodass je nach Signalwort nur die richtigen Ergebnisse ausgegeben werden. Siehe Abbildung 7 als Beispiel. Tabelle 2 zeigt welche Filter für welche Signalwörter verwendet werden.

(2) Im Teil A bzw. B wird keine Entität erwähnt, welche auch in der Lösungsmenge der Kandidaten vorkommt. Betrachte die Beispielfrage q = "Wer war der Präsident der Vereinigten Staaten vor (before) dem ersten Weltkrieg?". Diese wurde zuvor in die Teile Q = "Wer war der Präsident der Vereinigten Staaten?" und A = "dem ersten Weltkrieg?" aufgeteilt. Nun wird die Frage q(A) = "Wann begann der erste Weltkrieg?" an Aqqu gestellt. Sei c ein Kandidat für Q und c' der beste Kandidat von q(A). Um einen Kandidaten, der die Frage q beantwortet, zu erhalten, wird der Kandidat c um eine Relation erweitert. Diese erwartet als Objekt ein Datum. Nun werden die beiden Kandidaten zusammengeführt. Es wird ein Filter[2] entsprechend dem Signalwort eingefügt, sodass nur die richtigen Ergebnisse ausgegeben werden. Siehe dazu Tabelle 2.

(3) Im Teil A bzw. B wird ein Datum erwähnt. Dieses wird einfach als Knoten mit dem entsprechenden Filter in den Kandidaten eingefügt.

Da diese Signalwörter nicht immer zur zeitlichen Beschränkung verwendet werden, muss auch die originale, nicht aufgeteilte Frage dem System gestellt werden. Am Ende werden die Liste der Kandidaten zu beiden Fragen konkateniert und das Ranking wird berechnet.

Frage mit Signalwort	Zusätzlich gestellte Frage, falls Fall (2)	Filter
Q after A Q following A Q since A	Q1: Wann endete A	Fall 1: date(Q) > date(A) Fall 2: date(Q) > answer(Q1)
Q before A	Q1: Wann begann A	Fall1: date(Q) < date(A) Fall 2: date(Q) < answer(Q1)
Q during A Q while A Q for A	Q1:Wann begann A Q2: Wann endete A	Fall 1: Nicht möglich Fall 2: answer(Q1) <= date(Q) <= answer(Q2)
Q from A to B Q between A and B Q about A – B	Q1: Wann passierte A Q2: Wann passiere B	Fall 1: date(A) <= date(Q) <= date(B) Fall 2: answer(Q1) <= date(Q) <= answer(Q2)

Tabelle 2: Die Fragen die zusätzlich gestellt werden, wenn Fall (2) eintrifft. date(Q) ist das Objekt der hinzugefügten Datums-Relation. answer(Q) ist die Antwort auf die Frage Q. Das ist das Ergebnis des besten Kandidaten.

2 Mit der mir zur Verfügung gestellten Software zum Ausführen von SPARQL-Abfragen scheinen Datum-Filter nicht zu funktionieren

3.5 Superlativ-Fragen

Um Superlative zu erkennen werden die zuvor berechneten POS-Tags der Wörter verwendet. Superlative haben POS-Tag JJS oder RBS. Die Superlative, die gleichzeitig ein Signalwort für Temporal-Fragen darstellen, werden hier ignoriert. Soll ein Kandidat erweitert werden, so wird wie in Abschnitt 3.2 erklärt, zunächst ein Knoten für die Erweiterung gefunden. In diesem Fall ist eine Erweiterung am Zwischenknoten, an einem ehemaligen Antwortknoten und am aktuellen Antwortknoten sinnvoll. Nun wird der Kandidat, ähnlich wie bei Temporal-Fragen, um eine Relation erweitert, welche als Objekt o eine Zahl erwartet. Falls der Superlativ einer Relation nicht zugeordnet werden kann, so erweitere den Kandidaten nicht mit dieser Relation. Nun muss, je nach Superlativ, nur der Kandidat zurückgegeben werden, bei dem der Wert von o maximal oder minimal ist. Dies geschieht, wie bei Temporal-Fragen, durch sortieren an dem Objekt und setzten des LIMIT auf 1. Um zu bestimmen, ob bei einem gegebenen Superlativ das Maximum oder das Minimum ausgegeben werden soll, bin ich wie folgt vorgegangen. Weitere Möglichkeiten werden in Abschnitt 7 diskutiert.

Liste einiger Superlative: Eine von Hand erstellte Liste in der für die häufigsten Superlative steht, ob bei ihnen das Minimum oder das Maximum ausgegeben soll. Bei Superlativen, bei denen das von Kontext abhängt, wie beispielsweise „beste(r)“, hat diese Methode Probleme, genauso wie bei Superlativen, die nicht in der Liste vorkommen. Bei diesen habe ich festgelegt, dass das Maximum ausgegeben werden soll.

3.6 Features und Ranking

Da bei erweiterten Kandidaten mehr Wörter den Entitäten oder Relation zugeordnet werden können, sollten diese von Natur aus höher im Ranking stehen. Tabelle 3 zeigt alle neuen Features des Systems mit einer kleinen Erläuterung für jedes Feature.

ID	Beschreibung
25	Anzahl der Signalwörter, die zu Erweiterungen gemäß Abschnitt 3.4.1 (einfache Temporal Fragen) zu Erweiterungen geführt haben
26	Ob die Frage gemäß Abschnitt 3.4.2 als Frage mit expliziter zeitlicher Beschränkung interpretiert wurde.
27	Anzahl der Filter die gemäß Abschnitt 3.4.2 Fall 1 (Temporal Fragen mit expliziter zeitlicher Beschränkung – Entitäten in Lösungsmenge) hinzugefügt wurden
28	Anzahl der Filter die gemäß Abschnitt 3.4.2 Fall 2 (Temporal Fragen mit expliziter zeitlicher Beschränkung – Entitäten nicht in Lösungsmenge) hinzugefügt wurden
29	Anzahl der Filter die gemäß Abschnitt 3.4.2 Fall 3 (Temporal Fragen mit expliziter zeitlicher Beschränkung – Beschränkung mittels Datum) hinzugefügt wurden
30	Anzahl der Superlative, die wie in Abschnitt 2.5 erklärt, zu Erweiterungen geführt haben

Tabelle 3: Neue Features für das Ranking System von Aqqu

3.7 Erweiterungsprozess

In den vorherigen Abschnitten wurde erläutert wie die Erweiterung der Kandidaten funktioniert. In diesem Abschnitt wird nun erklärt, in welcher Reihenfolge ich die Erweiterungen ausgeführt werden.

1. Zunächst wird die Frage an den Signalwörtern für explizite Zeitliche Beschränkung, wie im Abschnitt 3.4 erläutert, aufgeteilt. Sowohl die aufgeteilte Frage als auch die nicht aufgeteilte Frage werden dem System nacheinander gestellt. Falls kein solches Signalwort in der Frage vorkommt, wird nur die nicht aufgeteilte Frage dem System gestellt.
2. Wie in Abschnitt 3.4 werden die Kandidaten zur Erweiterung bestimmt.
3. Es werden nacheinander die Erweiterungen von Abschnitt 3.3 (Zwei Fragen in einer), 3.4.1 (einfache Temporal-Fragen), 3.5 (Superlativ-Fragen) ausgeführt, falls sinnvoll. Die Menge der Kandidaten wird in jedem Schritt um die neuen Kandidaten erweitert.
4. Es wird für alle Kandidaten der aufgeteilten Frage die Erweiterung gemäß Abschnitt 3.4.2 (Temporal-Fragen mit expliziter zeitlicher Beschränkung) vorgenommen.
5. Das Ranking aller generierter Kandidaten wird berechnet und der beste Kandidat wird zurückgegeben

4. Verbessertes Answer Type Matching

Das ursprüngliche Aqqu-System [2] benutzte einen einfachen, aber durchaus effektiven Mechanismus für das Answer Type Matching (dt. Antworttyperkennung). Dieser berücksichtigte nur das Fragewort, welches bei den meisten Fragen das erste Wort darstellt. Wenn beispielsweise eine Frage gestellt wird, welche mit „Wer“ beginnt, so wurde eine Person, ein Charakter oder eine Organisation als Antwort erwartet. Das dies nicht immer der Fall ist, zeigt die Frage „Wer ist Johnny Depp?“. Hier wird eine Liste von Berufen erwartet. Die Problematik mit den „Wer ist ...“ Fragen wurde im originalen Aqqu-System mit den in Abschnitt 1.3 erwähnten N-Gram Relation Matching Feature gelöst. Allerdings ist ein verbessertes Answer Type Matching immer noch sehr sinnvoll. Besonders wichtig ist Answer Type Matching bei Fragen, bei denen das Zuordnen der Wörter zu Relationen nicht gut funktioniert. Das ist besonders bei Fragen der Fall, deren Verb „sein“ ist. Solche Fragen kommen in der Praxis häufig vor. Ein Kandidat, dessen Antworttyp mit dem erwarteten Antworttyp einer Frage übereinstimmte, bekam dann in einem speziellen Feature eine 1 als Wert, sodass er im Ranking höher eingeordnet wurde.

4.1 Answer Type Matching basierend auf Machine Learning

Das Answer Type Matching, welches ich im Folgenden vorstellen werde ist von [10] inspiriert und funktioniert mit den Entitätstypen von Freebase. Als Klassifizierer-Algorithmus habe ich Random Forest ausgewählt, da dieser die besten Ergebnisse in meinen Tests erzielt hat. Es handelt sich hierbei um einen Klassifizierer, der auf Machine Learning basiert. Deshalb muss jede Frage in einen Feature Vector (dt. Merkmalsvektor) umgewandelt werden. Hierbei reichen jedoch die Worte einer Frage als Features nicht aus, da noch weitere Informationen benötigt werden um gute Ergebnisse zu erzielen. Zunächst werden zu allen Worten die POS-Tags mit dem Stanford POS-Tagger [12] berechnet und die Worte werden damit annotiert. Das macht man aus dem Grund, dass es häufig auf die syntaktische Rolle des Wortes ankommt, anstatt auf das Wort selbst [11]. Die ersten Features sind also die annotierten Worte (1-grams) und 2-grams der annotierten Worte. Tabelle 4 zeigt diese Features für die Frage „Wer ist die Ehefrau des Präsidenten der Vereinigten Staaten?“

1-grams	Who.WP, is.VBZ, the.DT, wife.NN, of.IN, the.DT, president.NN, of.IN, the.DT, …
2-grams	Who.WP is.VBZ, is.VBZ the.DT, the.DT wife.NN, wife.NN of.IN, of.IN the.DT, ...

Tabelle 4: 1-gram und 2-gram Features für die Frage „Wer ist die Ehefrau des Präsidenten der Vereinigten Staaten“.

Als nächstes werden die NP-chunks (noun phrase chunks) und die VP-chunks (verb phrase chunks) der Frage betrachtet. Diese sind nicht-überlappende Teile der Frage, wie sie in [1] definiert wurden. Tabelle 5 zeigt die NP- und VP-chunks der Beispielfrage. Diese sind ebenfalls Features für den Klassifizierer. Um die chunks zu erhalten verwende ich [4].

NP-chunks	the.DT wife.NN, the.DT president.NN, the.DT united.NNP states.NNPS
VP-chunks	is.VBZ

Tabelle 5: NP- und VP-chunks der Frage „Wer ist die Ehefrau des Präsidenten der Vereinigten Staaten“.

Besondere Bedeutung für den gesuchten Antworttyp haben hierbei die jeweils ersten NP- und VP-chunks. Diese werden Head NP-chunk bzw. Head VP-chunk genannt. Der Head NP-chunk der Frage „Wer ist der höchste Berg“ ist „der höchste Berg“. Durch diesen kann man erkennen, dass die gesuchte Antwort ein Berg sein sollte. Ein weiteres verwendetes Features in diesem Zusammenhang ist das letzte Wort des Head NP-chunks, genannt Head Word. Siehe Beispiele für diese Features in Tabelle 6.

Head NP-chunks	the.DT wife.NN
Head VP-chunks	is.VBZ
Head Word	wife.NN

Tabelle 6: Head NP- und Head VP-chunk sowie Head Word der Frage „Wer ist die Ehefrau des Präsidenten der Vereinigten Staaten“.

Da es häufig darauf ankommt, was die Wörter bedeuten, muss es auch noch einige semantische Features geben. Das erste Beispiel für semantische Features sind alle Entitäten, welche von Aqqu identifiziert werden, als auch deren Typ. Besonders interessant sind hierbei die Entitäten im Head NP-chunk. Tabelle 7 zeigt alle Features im Zusammenhang mit den identifizierten Entitäten der Frage.

Entitäten	President of the United States, The Wife, United States of America, ...
Entitätstypen	government_office, book, location, …
Entitäten im Head NP-chunk	The Wife, ...
Entitätstypen im Head NP-chunk	Book, ...

Tabelle 7: Entitätsfeatures der Frage „Wer ist die Ehefrau des Präsidenten der Vereinigten Staaten“.

Zuletzt werden die WordNet [7] Senses der Wörter betrachtet. In WordNet werden Wörter in Senses organisiert. In einem WordNet Sense haben alle Wörter dieselbe Bedeutung. Sie sind also eine gute Möglichkeit um die Bedeutung der Wörter zu repräsentieren. Für jede Frage werden die WordNet Sense IDs, und die IDs der direkten Hyperonyme (Oberbegriffe) und Hyponyme (Unterbegriffe) der Wörter als Features verwendet. Hyperonyme und Hyponyme werden ebenfalls von WordNet zur Verfügung gestellt. Tabelle 8 zeigt die WordNet Features der Beispielfrage.

Sense IDs	president_of_the_united_states.n.01, president.n.04, …
Hyperonym IDs	head_of_state.n.01, spouse.n.01, …
Hyponym IDs	run_into.v.01, fall.v.16, diverge.v.02, reach.v.06

Tabelle 8: WordNet Features der Frage „Wer ist die Ehefrau des Präsidenten der Vereinigten Staaten“

4.2 Trainieren

Beim Trainieren von Aqqu wird ein nur ein Teil des gegebenen Trainingssets verwendet, um das Answer Type Matching System zu trainieren. Der andere Teil wird verwendet um es nach dem trainieren zu evaluieren und dabei wichtige Features zu berechnen. Wie man in der Evaluation in Abschnitt 5.4 erkennen kann, hat das Answer Type Matching System Probleme seltene Antworttypen zu erkennen. Mit den durch die Evaluation berechneten Features soll dem Ranking System mitgeteilt werden, wann eine Klassifizierung wahrscheinlich korrekt ist und wann nicht. Alle neuen Features für das Ranking System von Aqqu werden in Abschnitt 4.3 vorgestellt.

Bei Trainieren von Aqqu werden zunächst alle Kandidaten für die Fragen im Trainingsset generiert. Bei jeder Frage im Trainingsset werden außerdem die erwarteten Antworten mitgeliefert. Der beste Kandidat einer Frage ist der Kandidat, dessen Ergebnismenge am besten mit den erwarteten Antworten übereinstimmen. Sei q eine Frage und sei c der beste Kandidat für q. Für das Trainieren des Answer Type Matching Systems wird dann der erwarte Typ der Zielrelation von c als Antworttyp für q verwendet. Die Zielrelation eines Kandidaten ist die Relation, dessen Objekt der Antwortknoten ist. Die erwarteten Typen der Relationen werden im Voraus berechnet. Der Antworttyp eines Kandidaten ist entsprechend der erwartete Typ der Zielrelation des Kandidaten.

4.3 Features für das Ranking System von Aqqu

Nun muss die Feature-Repräsentation der Kandidaten noch um einige Features erweitert werden, damit die Kandidaten, bei denen der Antworttyp mit dem klassifizierten Antworttyp der Frage übereinstimmt, höher eingestuft werden können. Tabelle 9 zeigt die neuen Features mit Beschreibung. Features 31 und 32 werden bei der Evaluation berechnet und sollen dem Ranking System von Aqqu Informationen über die Qualität der Klassifizierung geben. Wie diese Werte berechnet werden, wird in Abschnitt 5.4 erklärt.

29	Antworttyp des Kandidaten stimmt mit dem klassifizierten Antworttyp überein
30	Wahrscheinlichkeit, dass der Antworttyp des Kandidaten der Antworttyp der Frage übereinstimmt. Diese wird vom Random Forest zur Verfügung gestellt.
31	Precision(c), wo c der klassifizierte Antworttyp ist. Das ist der Anteil der korrekt klassifizierten Fragen an den Fragen, welche mit c klassifiziert wurden.
32	Recall(c'), wo c' der Antworttyp des Kandidaten ist. Das ist der Anteil der korrekt klassifizierten Fragen an den Fragen, welche mit c' als Antworttyp haben.

Tabelle 9: Neue Features für das Ranking System von Aqqu.

5. Evaluation

In diesem Kapitel werden die vorgestellten Erweiterungen evaluiert und das resultierende System wird mit dem ursprüngliche System verglichen.

5.1 Daten

Als Wissensdatenbank verwende ich Freebase mit seinen 2,9 Milliarden Fakten und 44 Millionen Entitäten. Als Benchmarks verwende ich die Datensätze WebQuestions [3] und den Free917 [5]. Beide beinhalten Fragen mit den zugehörigen Antworten. Jedoch unterscheiden sich die Art der Fragen der beiden Datensätze stark. Da die beiden Datensätze jedoch fast keine komplexen Fragen beinhalten, verwende ich noch einen eigenen Datensatz an Fragen als Benchmark.

Free917: Der Free917 Datensatz beinhaltet 917 Fragen aus insgesamt 81 Bereichen [5]. Die Fragen beinhalten im Durchschnitt 6,3 Worte pro Frage [5]. Die Übersetzungen decken 635 unterschiedliche Relationen ab [5]. Jeder Bereich deckt maximal 6% der Fragen ab [5]. Die beiden Bereiche mit den häufigsten Fragen sind *Film* und *Business*. „What are the neigborhoods in New York City?“ und „How many countries use the Rupee?“ sind Beispielfragen dieses Datensatzes. Die Fragen wurden von zwei Muttersprachlern angefertigt und sind grammatikalisch korrekt [5]. Die meisten Fragen im Datensatz sind mit Freebase beantwortbar.

WebQuestions: WebQuestions beinhaltet 5.810 Fragen, die durch Crawling der Google Suggest API erhalten wurden [3]. Dadurch sind sehr viele Fragen enthalten, die sehr oft in Google gefragt wurden. Im Gegensatz zu Free917 sind die Fragen oft grammatikalisch nicht korrekt. Des Weiteren sind die Fragen im Datensatz oft sehr einfach und beinhalten meist nur eine Entität [3]. Die Fragen „Where did Jackie Kennedy go to college?“ und „What is spoken in Czech Republic?“ sind Beispielfragen für WebQuestions. Die Antworten wurden mittels Crowdsourcing ermittelt [3]. Deshalb sind die Antworten nicht immer ganz korrekt. Manchmal fehlen einige Elemente der korrekten Antwortmenge.

Eigener Datensatz: Der Datensatz enthält 40 komplexe Frage. Jeweils 10 Fragen für jeden der folgenden Typen: (1) Zwei Fragen in einer – Typ 1, (2) Zwei Fragen in einer, Typ 2, (3) einfache

Temporal-Fragen und (4) Superlativ-Fragen. (1) und (2) wurden in Abschnitt 3.3 erklärt, (3) in Abschnitt 3.4.1 und (4) in Abschnitt 3.5. Die Fragen sind im Anhang dieser Arbeit zu sehen.

5.2 Evaluation der Erweiterungen zur Beantwortung komplexer Fragen

In diesem Abschnitt werden alle Erweiterungen evaluiert, die das Beantworten komplexer Fragen ermöglichen sollen. Diese wurden in Kapitel 3 vorgestellt.

5.2.1 Daten

Sowohl WebQuestions als auch Free917 enthalten sehr wenige komplexe Fragen. Deshalb verwende ich meinen eigenen Datensatz zur Evaluation. Da die Anzahl der Fragen in diesem Datensatz nicht ausreicht um Aqqu[2] zu trainieren, verwende ich das Trainingsset von Free917 dafür. Dieses beinhalten 70% (641) der Fragen. Beim Trainieren werden keine erweiterten Kandidaten generiert.

5.2.2 Evaluationsmaße

Zur Evaluation verwende ich die Accuracy.

Seien $q_{1,}q_{2,}\ldots,q_n$ die Fragen des Testsets. Seien $g_{1,}g_{2,}\ldots,g_n$ die korrekten Antworten für die Fragen (gold answers). Seien $a_{1,}a_{2,}\ldots,a_n$ die vom erweiterten System berechneten Antworten. Das ist jeweils die Antwortmenge des Kandidaten, der im Ranking ganz oben steht.

Accuracy ist der Anteil der Fragen, bei denen die berechnete Antwort exakt der gegebenen Antwort entspricht.

$$\text{Accuracy} = \frac{1}{n} \cdot \sum_{i=1}^{n} I(g_i = a_i)$$

Hierbei ist I(e) eine Indikatorfunktion, die eins zurückgibt, falls der Ausdruck e wahr ist und ansonsten null. Dieses Maß ist deshalb sinnvoll, da die korrekten Antworten auf die Fragen im Testset bekannt sind.

5.2.3 Ergebnisse

In Tabelle 10 sieht man die Ergebnisse aufgeteilt nach Fragetyp. Die genauen Ergebnisse sind im Anhang zu finden.

	Accuracy
Insgesamt	50 %
Zwei Fragen in einer – Typ 1	70 %
Zwei Fragen in einer – Typ 2	70 %
Einfache Temporal-Fragen	50 %
Superlativ Fragen	10 %

Tabelle 10: Ergebnisse der Evaluation mit dem eigenen Datensatz

Wie man sieht funktioniert das System am besten für Fragen von Typ „Zwei Fragen in einer“. Bei einfachen Temporal-Fragen und Superlativ Fragen sind die Ergebnisse dagegen schlechter. Dies kann daran liegen, dass die Superlative und Signalwörter zum Teil schlecht den Relationen zugeordnet werden können. Bei Superlativ Fragen wird in 70% der Fälle zumindest der Kandidat generiert. Die durchschnittliche Position des richtigen Kandidaten im Ranking beträgt hier 11,9. Bei Temporal-Fragen wurde bei 80% der Fragen der richtige Kandidat generiert. Die durchschnittliche Position der richtigen Kandidaten im Ranking beträgt 1,6.

5.3 Auswirkungen auf die Übersetzung einfacherer Fragen

Durch die Veränderungen werden mehr Kandidaten generiert. Es ist offensichtlich, dass die Qualität der Übersetzung einfacher Fragen abnimmt. In diesem Abschnitt wird evaluiert wie stark diese Verschlechterung ausfällt.

5.3.1 Daten

In diesem Fall trainiere ich wieder das System mit dem Free917 Datensatz. Einmal trainiere ich das System mit eingeschalteten Erweiterungen und einmal ohne. Ich verwende 70% der Fragen (641 Fragen) als Trainingsset und 30% (276 Fragen) als Testset.

5.3.2 Evaluationsmaße

Free917 enthält für jede Frage eine SPARQL-Abfrage, die die Übersetzung der Frage darstellt. Führt man diese auf Freebase aus, so enthält man die gesuchte Antwort. Diese ist immer vollständig und korrekt. Deshalb ist die Accuracy als Evaluationsmaß sinnvoll. Diese berücksichtigt nur komplett richtige Antworten. Sie wurde bereits in Abschnitt 5.2.2 definiert.

5.3.3 Ergebnisse

	Mit Erweiterungen trainiert. Mit Erweiterungen evaluiert.	Ohne Erweiterungen trainiert. Mit Erweiterungen evaluiert.	Ganz ohne Erweiterungen
Accuracy	57,2%	40,2%	65,9% [2]

Tabelle 11: Accuracy des Systems mit dem Free917 Datensatz.

Wie man in Tabelle 11 sehen kann, ist die Accuracy des Systems mit Erweiterungen signifikant schlechter. Durch die Erweiterungen werden deutlich mehr Kandidaten generiert. Da in Free917 jedoch kaum Fragen gestellt werden, die eine Erweiterung benötigen, kommt es nun häufiger dazu, dass ein falscher Kandidat im Ranking ganz oben steht. Trainiert man das System mit Erweiterungen, dann lernt es Kandidaten, die durch Erweiterungen entstanden sind, auszusortieren. Mögliche Verbesserungen werden in Abschnitt 7 diskutiert.

5.4 Evaluation Answer Type Matching

In diesem Abschnitt wird das verbesserte Answer Type Matching evaluiert, wie es in Kapitel 4 vorgestellt wurde.

5.4.1 Daten

Zur Evaluation des Answer Type Matching Systems verwende ich die Datensätze WebQuestions und Free917. Bei Free917 verwende ich 70% der Fragen (641 Fragen) als Trainingsset und 30% (276 Fragen) als Testset. Bei Free917 sind die gesuchten SPARQL-Abfragen schon gegeben. Für die Evaluation ist der gesuchte Antworttyp einer Frage von Free917 der erwartete Typ der Zielrelation der gegebenen SPARQL-Abfrage. Bei WebQuestions verwende ich 70% (3778 Fragen) zum Trainieren und 30% (2032 Fragen) zum Testen. Bei den Fragen von WebQuestions sind keine

SPARQL-Abfragen gegeben. Stattdessen ist die gesuchte Antwort gegeben. Der Antworttyp der Frage ist der Typ der Entitäten in der gesuchten Antwort.

5.4.2 Evaluationsmaße

Sei Q die Menge aller Fragen und sei Q' die Menge der Fragen, bei denen der Antworttyp korrekt identifiziert wurde. Accuracy ist der Anteil der korrekt klassifizierten Fragen:

$$\text{Accuracy} = \frac{Q'}{Q}$$

Sei D_c die Menge aller Fragen mit Antworttyp c. Sei D_c' die Menge aller Fragen, für die der Klassifizierer c als Antworttyp bestimmt hat. Die Maße Precision und Recall hängen von der jeweiligen Klasse ab.

$$\text{Precision}(c) = \frac{|D_c \cap D_c'|}{|D_c'|} \qquad \text{Recall}(c) = \frac{|D_c \cap D_c'|}{|D_c|}$$

Average Precision ist der Durchschnitt der Precision über alle Klassen. Average Recall analog. Sei n die Anzahl der Antworttypen c, für die Precision(c) definiert ist. Sei l die Anzahl der Antworttypen c, für die Recall(c) definiert ist.

$$\text{Average Precision} = \frac{1}{n} \cdot \sum_c \text{Precision}(c) \qquad \text{Average Recall} = \frac{1}{l} \cdot \sum_c \text{Recall}(c)$$

5.4.3 Ergebnisse

Tabelle 12 zeigt die Ergebnisse der Antworttyp-Klassifizierung. Mit einer Accuracy von 62% bei WebQuestions und 53% bei Free917 werden zumindest mehr als die Hälfte aller Fragen richtig klassifiziert. Jedoch sind die Werte für die Average Precision und den Average Recall mit unter 40% bescheiden. Dies liegt insbesondere daran, dass der Answer Type Matching System mit dem jeweils gegebenen Trainingsset nur die einfachsten und häufigsten Antworttypen bestimmen kann. Bei WebQuestions konnten 78 von 116 Antworttypen im Trainingsset nie korrekt bestimmt werden. Bei Free917 sind es 28 von 48. Jedoch sind die Precision und Recall Werte für viele häufige Antworttypen, wie people.person oder location.location gut.

Datensatz	WebQuestions	Free917
Accuracy	0,63	0.54
Average Precision	0,398	0.25
Average Recall	0,18	0.35
Beispiele für gute Precision Werte	food.ingedient: 1.0 military.military_conflict: 1.0 music.album: 1.0 people.person: 0.77 finance.currency: 0.76	tv.tv_theme_song: 1.0 film.producer: 1.0 religion.religion:1.0 type.int: 1.0 location.location: 0.667
Beispiele für gute Recall Werte	food.ingredient: 1.0 language.human_language: 0.93 location.location: 0.88 people.person: 0.84 finance.currency: 0.67	type.datetime; 1.0 location.location 1.0 film.director: 1.0 type.float: 0.77 type.text: 0.6
Anzahl Antworttypen, die nie korrekt bestimmt wurden	78 (insgesamt 116 Antworttypen im Testset)	28 (insgesamt 48 Antworttypen im Test)

Tabelle 12: Ergebnisse des Antworttype Klassifizierer

6. Effizienz

Die Laufzeit von Aqqu ohne Erweiterungen und Aqqu mit Erweiterungen verhält sich auf dem mir zur Verfügung gestellten Computer wie folgt: Ohne Erweiterungen dauert die Übersetzung einer Frage im Schnitt 871ms und mit Erweiterungen 1795ms. Besonders lange dauert die Identifizierung der Entitäten. Auch das Berechnen des Rankings kostet einige Zeit. Dieses muss beim erweiterten System zwei mal ausgeführt werden.

7. Zukünftige Arbeit

In diesem Abschnitt werden nun die möglichen Verbesserungen und Erweiterungen diskutiert.

7.1 Verbesserungen in der Erweiterung der Kandidaten

- Die Qualität der Übersetzung einfacher Fragen hat drastisch unter den Erweiterungen gelitten. Um diese wieder zu Verbessern wäre folgendes möglich: (1) Features der Frage, wie beispielsweise Anzahl der Worte, auch im Ranking Prozess von Aqqu zu verwenden. Dann könnten Kandidaten mit vielen Relationen bei Fragen mit wenigen Worten, weniger hoch im Ranking eingeordnet werden. Auch ein Klassifizierer, welcher entscheidet, ob eine Frage überhaupt erweitert werden soll, wäre denkbar. Geschätzte Zeit: Wenige Wochen.
- Die Qualität der Übersetzung von Superlativ- und einfachen Temporal-Fragen sind noch nicht zufriedenstellend. Hier müsste man noch einige Änderungen vornehmen. Man könnte zum Beispiel versuchen die Scores der Zuordnung von Superlativ bzw. Signalwort zu erhöhen. Auch des Trainieren mit Datensätzen, welche Superlativ- und Temporal-Fragen enthalten, könnte die Qualität der Übersetzung dieser Fragen verbessern. Geschätzte Zeit: Wenige Wochen.
- In Abschnitt 3.2 wurde erklärt wie die Knoten eines Kandidaten bestimmt wurden, an denen eine Erweiterung stattfinden soll. Diese wurden im Voraus, je nach Fragetyp, bestimmt. Ein Klassifizierer könnte zur Laufzeit besser bestimmen an welche Knoten ein gegebener Kandidat erweitert werden sollte. Dadurch würden dann womöglich weniger Kandidaten erzeugt werden. Das könnte sowohl Laufzeit als auch die Qualität des Systems verbessern. Geschätzte Zeit zum Implementieren dieses Klassifizierers: Wenige Wochen.
- Bisher wurden die Kandidaten, welche erweitert werden sollen, so bestimmt: Das Ranking wurde berechnet und die besten 10 Kandidaten wurden in der Erweiterung berücksichtigt (Abschnitt 3.1). Das Hauptproblem daran ist, dass das Berechnen des Rankings viel Laufzeit kostet. Eine verbesserter Mechanismus, eventuell ein auf Machine Learning basierender Klassifizierer, könnte hier die Laufzeit deutlich verbessern. Geschätzte Zeit zum Implementieren dieses Klassifizierers: Wenige Wochen.
- Manche Fragen enthalten die in Abschnitt 3.4 spezifizierten Signalwörter, ohne dass es sich bei der Frage um eine Temporal-Frage handelt. Falls eine Frage ein Signalwort für explizite zeitliche Beschränkung enthält, so wird die Frage in mehrere Teile aufgeteilt. Dies kostet Laufzeit. Falls es sich bei der Frage jedoch nicht um eine Temporal-Frage handelt, so ist das verschenkte Laufzeit. Ein Klassifizierer, der bestimmt ob es sich bei der Frage um eine Temporal-Frage handelt wäre also durchaus sinnvoll. Geschätzte Zeit: Wenige Wochen.
- Die in Abschnitt 3.4 spezifizierten unterstützten Signalwörter sind noch unvollständig. Es

fehlen beispielsweise „when“, „on“, „in“, „at the time of“. Diese benötigen etwas komplexere SPARQL-Abfragen. Geschätzte Zeit: Wenige Wochen.

- Ein Problem bei den Superlativ-Fragen ist, dass bei manchen wenigen Superlativ-Fragen nicht klar ist, ob sie nach einem Maximum oder einem Minimum fragen. Die Frage „Wer hat die beste Runde im Hockenheimring gefahren?“ fragt nach dem Fahrer mit der niedrigsten Fahrzeit. In der Regel fragt „beste“ jedoch nach Maxima. Auch hier könnte ein Klassifizierer weiterhelfen. Dafür müsste jedoch auch ein geeignetes Trainingsset erstellt werden. Geschätzte Zeit: Wenige Wochen.
- Das erweiterte System ist immer noch nicht in der Lage noch komplexere Fragen zu beantworten. Beispiele hierfür wären „3 oder mehr Fragen in einer“, „komplexe Superlative“ wie „Welcher Schauspieler kommt in den meisten Filmen vor?“, oder „Temporal Fragen mit Personalpronomen“, wie „Welchen Beruf hatte Barack Obama, bevor er Präsident wurde?“. Geschätzte Zeit zum Erweitern des Systems, dass auch solche Fragen beantwortet werden können: Wenige Monate.

7.2 Verbesserungen des Answer Type Matching

- Das Hauptproblem des vorgestellten Answer Type Matching Systems ist, dass es viel zu viele mögliche Antworttypen gibt. Ein Reduzierung der Anzahl der Antworttypen bzw. Verwendung eigener Antworttypen könnte das System deutlich verbessern. Geschätzte Zeit: Ein paar Wochen.
- Andere Answer Type Matching Systeme sind hierarchisch aufgebaut. Das hier vorgestellte ist dies nicht. Vermutlich würde eine Umwandlung in ein hierarchisches System eine Verbesserung hervorrufen. Geschätzte Zeit: Einige Wochen.

7.3 Verbesserungen an der Evaluation

- Die vorhandenen Datensätze Free917 [5] und WebQuestions [3] sind beide nicht geeignet um das erweiterte System richtig zu evaluieren. Der kleine Datensatz, der hier zur

Evaluation verwendet wurde, enthält viel zu wenige Fragen. Es müsste ein neuer Datensatz mit ausreichend vielen komplexen Fragen angelegt werden.

- Ich war außerdem nicht in der Lage die Evaluationen mit dem großen WebQuestions Datensatz durchzuführen. Insbesondere eine Evaluation der Auswirkung des verbesserten Answer Type Matching Systems wäre sinnvoll. Dies ist mit WebQuestions sinnvoller als mit Free917, da bei Free917 zu wenige Fragen für die Evaluation des Answer Type Matching Systems beim Trainieren von Aqqu übrig bleibt. Geschätzte Zeit: eine Woche.

8. Fazit

Das vorgestellte System ist definitiv in der Lage einen gegebenen Kandidaten so zu erweitern, dass er eine komplexe Frage beantwortet. Jedoch muss noch daran gearbeitet, dass nicht zu viele Kandidaten erzeugt werden. Insbesondere muss ein neuer Datensatz an Fragen angelegt werden, mit dem das System vernünftig trainiert und evaluiert werden kann. Ist das getan, kann ich mir durchaus vorstellen, dass das erweiterte System gute Ergebnisse im beantworten komplexer Fragen erziehlen kann, ohne zu große Einbußen bei einfachen Fragen zu erleiden.

Das Answer Type Matching System ist mit einer Accuracy von etwa 60% beim WebQuestions [3] Datensatz auf jeden Fall einen Schritt nach vorne. Natürlich muss es noch verbessert werden, um an die Qualität anderer Answer Type Matching Systeme heranzukommen. Insbesondere muss die Anzahl der möglichen Antworttypen verringert werden, da das Answer Type Matching System momentan nicht in der Lage ist Fragen mit seltenen oder komplizierten Antworttypen richtig zu klassifizieren.

9. Referenzen

[1] S. P. Abney. Parsing by chunks. In S. P. Abney R. C. Berwick and C. Tenny, editors, Principle-based parsing: Computation and Psycholinguistics. Kluwer, Dordrecht, Seiten 257–278, 1991.

[2] H. Bast, E. Haussmann. More Accurate Question Answering on Freebase. In CIKM, Seiten 1431–1440, 2015

[3] J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic Parsing on Freebase from Question-Answer Pairs. In EMNLP, Seiten 1533–1544, 2013.

[4] S. Bird, Ewan K, E. Loper. Natural Language Processing with Python, Kapitel 7

[5] Q. Cai and A. Yates. Large-scale Semantic Parsing via Schema Matching and Lexicon Extension. In ACL, Seiten 423–433, 2013.

[6] ClueWeb, 2012. The Lemur Projekt

[7] C. Fellbaum. WordNet. Wiley Online Library, 1998.

[8] S. He, S. Liu, Y. Chen, G. Zhou, K. Liu, and J. Zhao. CASIA@QALD-3: A Question Answering System over Linked Data. In CLEF (Working Notes), 2013

[9] V. Krishnan, S. Das and S. Chakrabart. Enhanced Answer Type Inference from Questions using Sequential Models. In HLT/EMNLP. ACL, Seiten 315–322, 2005

[10] X. Li, and D. Roth. Learning Question Classifiers: The Role of Semantic Information. NLE 12. Jg. Nr. 03, Seiten 229–249, 2006

[11] X. Li, K. Small, and D. Roth. 2004. The role of semantic information in learning question classifiers. In IJCNLP, 2004

[12] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky. The stanford corenlp natural language processing toolkit. In ACL, Seiten 55–60, 2014.

[13] E. Saquete, J. L. Vicedo, P. Martínez-Barco, R. Muñoz, H. Llorens, Enhancing QA Systems with Complex Temporal Question Processing Capabilities. In JAIR 35, Seiten 775–881, 2009

[14] C. Unger, C. Forascu, V. Lopez, A. N. Ngomo, E. Cabrio, P. Cimiano, and S. Walter. Question, answering over linked data (QALD-4). In CLEF 2014, Seiten 1172–1180, 2014

[15] V. I. Spitkovsky and A. X. Chang. A Cross-Lingual Dictionary for English Wikipedia Concepts. In LREC, Seiten 3168–3175, 2012.

[16] M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath , V. Tresp , G. Weikum. Deep Answers for Naturally Asked Questions on the Web of Data. In WWW, Seiten 445–449, 2012

[17] P. Yin, N. Duan, B. Kao, J. Bao, and M. Zhou. Answering Questions with Complex Semantic Constraints on Open Knowledge Bases. In CIKM, Seiten 1301–1310, 2015

[18] L. Zou, R. Huang, H. Wang, J. Xu Yu, W. He, and D. Zhao. Natural Language Question Answering over RDF – A Graph Data Driven Approach. In SIGMOD, Seiten 313–324, 2014

10. Anhang

10.1 Eigener Datensatz zur Evaluation mit genauen Ergebnissen

10.1.1 Superlativ-Fragen

Who was the tallest president of the United States?
→ Richtiges Ergebnis auf Rang 9.
→ Features ausgeglichen im Vergleich zum ersten.
→ „Who is the tallest president of the United States of America?“ funktioniert.

What is the largest country in Europe?
→ Richtiges Ergebnis auf Rang 18.
→ Rang 1: „largest“ wird mit hohem Score der Relation „contains_major_portion_of“ zugeordnet.

Which movie directed by Michael Bay has the longest runtime?
→ Problem: „runtime“ ist eine k-äre (k>2) Relation, die noch nicht unterstützt wird.

Which player of the german national team has the most weight?
→ Richtiges Ergebnis auf Rang 1.
→ Rang 1 hat mehr Literal Matches.

What is the highest mountain in Asia?
→ Kandidat nicht generiert.
→ „What are mountains in Asia?“ sind zwei Fragen in einer. Diese Frage funktioniert auch ohne Superlativ nicht.

What is the highest building in Germany?
→ Kandidat nicht generiert.
→ „What are buildings in germany?“ sind zwei Fragen in einer. Diese Frage funktioniert auch ohne Superlativ nicht.

Who was the tallest inventor of the light bulb?
→ Richtiger Kandidat auf Rang 4.
→ Den Features zufolge sollte eigentlich der richtige Kandidat auf Rang 1 sein.

What is the highest mountain in the Alps?
→ Richtiger Kandidat Rang 3.
→ Features ausgeglichen in Vergleich zu Rang 1.

What is the highest mountain in Himalaya?
→ Richtiger Kandidat Rang 3.
→ Ausgeglichene Features im Vergleich zu Rang 1.

Who is the tallest player of the german national football team?
→ Richtiger Kandidat Rang 45.
→ In Rang 1 wurden mehr Wörter durch einen Literal Match einer Relation zugeordnet.

10.1.2 Temporal-Fragen

Which inventor of the light bulb died first?
→ Richtiges Ergebnis auf Rang 1.
→ In Rang 1 wurden mehr Wörter durch einen Literal Match einer Relation zugeordnet.

What is the first James Bond book?
→ Richtiges Ergebnis auf Rang 1

What is the last James Bond book?
→ Richtiges Ergebnis auf Rang 2.
→ Rang 1 hat besseren Score bei einem Synonym-Match.

What is the first book of Kafka?
→ Richtiges Ergebnis auf Rang 1.

What is the first movie of Michael Bay?
→ Richtiges Ergebnis auf Rang 1.

What was the first game developed by Nintendo?
→ Richtiges Ergebnis auf Rang 2.
→ In Rang 1 wurden mehr Wörter durch einen Literal Match einer Relation zugeordnet.

Who is currently the president of the United States?
→ Richtiges Ergebnis auf Rang 4.
→ Features ausgeglichen im Vergleich zu Rang 1.

What is the current population of Berlin?
→ Richtiges Ergebnis auf Rang 1.
→ In Rang 1 wurden mehr Wörter durch einen Literal Match einer Relation zugeordnet.

Who is the current chancellor of Germany?
→ Kandidat wird nicht erzeugt oder im Zuge des Pruning entfernt.
→ Problem: „chancellor of Germany“ wird der Entität Angela Merkel zugeordnet.

Who is the current trainer of the German national football team?
→ Kandidat wird nicht erzeugt oder im Zuge des Pruning entfernt.

10.1.3 Zwei Fragen in einer – Typ 1

What are the developers of games published by Electronic Arts?
→ Richtiges Ergebnis auf Rang 1.

What are the capitals of the countries in Europe?
→ Richtiges Ergebnis auf Rang 1.
→ Summer der Scores der Context-Matches ist bei Rang 1 besser.

Who is the wife of the president of the United States?
→ Richtiges Ergebis auf Rang 1.

Who played in a movie directed by Michael Bay?
→ Richtiges Ergebnis auf Rang 6.
→ In Rang 1 wurden mehr Wörter durch einen Synonym Match einer Relation zugeordnet.

What is the area of the capital of germany?
→ Richtiges Ergebnis auf Rang 1.

Who is wife of the inventor of the light bulb?
→ Richtiges Ergebnis auf Rang 1.

When is the inventor of the lightbulb born?
→ Richtiges Ergebnis auf Rang 1.

Who are the parents of the CEO of facebook.
→ Richtiges Ergebnis auf Rang 1?

Which presenter created Wetten dass?
→ Richtiges Ergebnis auf Rang 5.
→ In Rang 1 wurden mehr Wörter durch einen Literal Match einer Relation zugeordnet.

What is the profession of the CEO of Telekom?
→ Richtiger Kandidat wird nicht erzeugt → „the CEO“ wird Entität „Edvard McVaney“ zugeordnet
→ „What is the profession of CEO of Telekom?“ → Rang 1

10.1.4 Zwei Fragen in einer – Typ 2

Which games developed by Crytec are published by Electronic Arts?
→ Richtiges Ergebnis auf Rang 1

Which countries are in Europa and in Asia?
→ Richtiger Kandidat im Prinzip auf Rang 1, allerdings unerwartete Ergebnismenge.

Who was the husband of Rachel Jackson and president of the United States?
→ Richtiges Ergebnis auf Rang 1.

In which movie directed by Michael Bay played Will Smith?
→ Richtiges Ergebnis auf Rang 1.

Who is founder of SAP and patron of TSG Hoffenheim?
→ Richtiges Ergebnis auf Rang 1.

Which actor from Germany won an Oscar?
→ Kandidat nicht generiert (einzelne Fragen funktionieren selbst nicht gut).

Who acted in Bad Boys 2 and Prince of Bel Air?
→ Richtige Antwort Rang 14.
→ Rang 1 allgemein höhere Scores.

Who is son of Edward Zuckerberg and CEO of Facebook?
→ Richtiger Kandidat Rang 1
→ CEO of Facebook wäre schon eindeutig, jedoch wird auch der zur Frage passende Kandidat erzeugt.

Who won an Oscar and was born in Austria?
→ Fast richtiger Kandidat (Oscar in Nebenrolle, geboren in Österreich) auf Rang 2.

Which player of the german national football team was born in Gliwice?
→ Richtiger Kandidat auf Rang 1.