Bachelorarbeit

Semantische Suche
-
Entitätenerkennung

Ina Baumgarten

4. Oktober 2011

Albert-Ludwigs-Universität Freiburg im Breisgau
Technische Fakultät
Institut für Informatik
Lehrstuhl für Algorithmen und Datenstrukturen

Bearbeitungszeitraum
11. 07. 2011 –11.i̇0. 2011

Gutachterin
Prof. Dr. Hannah Bast

Betreuer
Prof. Dr. Hannah Bast
Björn Buchhold

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.

Das in dieser Arbeit beschriebene Projekt wurde in Zusammenarbeit mit Niklas Meinzer entwickelt, welcher ebenfalls über diesen Themenbereich eine Bachelorarbeit verfasst hat. Die unterschiedlichen Teilbereiche wurden jedoch selbstständig bearbeitet. Alle in dieser Arbeit beschriebenen Themen wurden von mir bearbeitet. Sollten zugunsten der Vollständigkeit bearbeitete Themen von Niklas Meinzer erwähnt werden, wird dies im Text explizit gekennzeichnet.

Inhaltsverzeichnis

Zusammenfassung **1**

Abstract **3**

1 Einleitung **5**
1.1 Semantische Volltextsuche 7
1.1.1 Grundlagen 7
1.1.2 Bisherige Arbeit 8
1.1.3 Problematik für deutsche Zeitungsartikel 8
1.2 Zielsetzung 9
1.3 Beitrag der Arbeit 9

2 Vorverarbeitung **11**
2.1 Parsen der deutschen Wikipedia 11
2.1.1 Weiterleitungen 12
2.1.2 Disambiguierungsseiten 12
2.2 Parsen der Artikel 12
2.2.1 Part-Of-Speech Tagging 12
2.2.2 Unwichtige Wörter 13
2.2.3 Zählen der Entitäten 13
2.3 Parsen der YAGO-Facts 14

3 Methoden der Entitätenerkennung **17**
3.1 PosTagging 17
3.2 Stemming 17
3.3 Kompositazerlegung 18

4 Entitätenerkennung **19**
4.1 Entitäten 20
4.2 Datenstruktur 20
4.2.1 Aufbau 20
4.2.2 Modifikationen 21
4.3 Regeln 22
4.4 Multientitätenerkennung 24
4.5 Teilentitätenerkennung 26
4.5.1 Algorithmus 26

4.5.2 Scoring 28

5 Evaluation **31**
5.1 Aufbau einer Ground Truth 31
5.2 Evaluation einer Ground Truth 32

6 Resultate und Diskussion **35**
6.1 Interpretation der Evaluation 35
6.2 Unmögliche Entitäten 36
6.3 Genauigkeit 37
6.4 Trefferquote 38
6.4.1 Unzureichende Kontexterkennung 38
6.4.2 Kettenreaktionen 39
6.4.3 Weitere Probleme 40

7 Ausblick **43**

Danksagung **47**

Literaturverzeichnis **49**

Zusammenfassung

In dieser Bachelorarbeit wird eine Entitätenerkennung für deutsche Texte vorgestellt, die eine bereits implementierte semantische Suchmaschine an der Universität Freiburg erweitert. Entitäten sind dabei als eindeutige Bedeutung von Wörtern definiert und werden unter Verwendung eines Tries gespeichert. Um das Ziel einer funktionierenden Entitätenerkennung zu erreichen, wurde Vorwissen, bestehend aus einer Menge von Entitäten, Artikeln und Füllwörtern generiert. Mit Hilfe des Vorwissens kann anschließend eine Entitätenerkennung durchgeführt werden, die sich in zwei wesentliche Schritte unterteilen lässt: die Erkennung eindeutiger Entitäten und die Erkennung uneindeutiger Entitäten. Zusätzlich werden verschiedene Verfahren verwendet, um die Resultate zu verbessern: Scoring, PosTagging, Stemming und Kompositazerlegung. Die Entitätenerkennung weist mit einer Trefferquote von 82% und einer Genauigkeit von 97% Entitäten die korrekte Bedeutung zu. Ausgeschlossen sind hierbei 19% von unmöglichen Entitäten, für die keine passende Entität existiert. Zukünftig kann die Entitätenerkennung mittels verschiedener Modifikationen noch weiter verbessert werden.

Abstract

This paper introduces an entity recognition for german articles, which extends the functionality of an already implemented semantic search engine at the University of Freiburg. Entities are defined as unique meaning of words and are stored by using a trie. For achieving a working entity recognition, previous knowledge was generated, consisting of a set of entities, articles and stopwords. Using the previous knowledge the entity recognition is performed in two essential steps: recognition of unambiguous entities and recognition of ambiguous entities. Moreover several methods are used for improving the results: scoring, postagging, stemming and decompounding. The entity recognition assigns with a sensitivity of 82% and an accuracy of 97% the correct meaning to an entity. Excluded are impossible entities, for which no fitting entity exist. In the future it is possible to improve the entity recognition by several modifications.

1 Einleitung

Volltextsuchmaschinen, wie Google, helfen gewünschte Informationen gezielt im World Wide Web zu finden. Wir geben Begriffe in die Suchmaschine ein und daraufhin werden uns alle Seiten ausgegeben, in welchen diese Suchbegriffe genau oder in leichter Abwandlung vorkommen.
Man nehme als Beispiel die Suche nach „Wii Kosten“. Die Volltextsuche würde nun alle Seiten auflisten, welche die Wörter „Wii“ und „Kosten“ oder „kostet“, etc. enthalten. Die Volltextsuche eignet sich sehr gut, um Informationen zu einem Fakt, wie „Wie hoch ist der Eiffelturm?“, zu finden. Ein Suchbeispiel hierfür wäre „Höhe Eiffelturm“. Selbiges gilt für Themenbereiche, wie „Was gibt es zum Thema Kalter Krieg?“. Nach der Eingabe von „Kalter Krieg“ erhält man sofort alle relevanten Suchergebnissen zu diesem Thema. Der Grund für den Erfolg von Volltextsuchmaschinen besteht darin, dass in vielen Fällen die gewünschten Informationen auf einer Seite stehen, welche vorhersehbare Begriffe, wie z.B. „Eiffelturm“ enthalten. Dadurch und mithilfe von Scoringalgorithmen, die den gefundenen Seiten je eine passende Relevanz zuordnen, können die Informationen meist mit nur wenigen Begriffen in den ersten Suchergebnissen gefunden werden. In manchen Fällen eignet sich die Volltextsuche sogar, um Listen, wie „alle Nobelpreisträger“, zu finden. In den meisten Fällen existieren solche Listen jedoch nicht, oder sie sind unvollständig oder fehlerhaft, da sie nicht gepflegt werden.
Betrachte man die Suche nach „Milliardäre spenden“. In diesem Fall möchte man nicht nach dem Wort „Milliardäre“ suchen, sondern nach jenen Milliardären, die spenden. Gibt man diese Suchbegriffe bei Google ein, so erhält man zahlreiche Artikel, die genau jene zwei Wörter enthalten. Eine passende Liste existiert nicht. Die gesuchten Milliardäre lassen sich somit mit der Volltextsuche nur dann direkt finden, wenn dem Suchenden diese Milliardäre bereits namentlich bekannt sind, was schlussendlich bedeutet, nach etwas zu suchen, was man bereits weiß.
Diese Bachelorarbeit beschäftigt sich mit einem alternativen Ansatz, der dieses Dilemma zu lösen versucht: der „Semantischen Volltextsuche“. So ist mit Milliardär nicht das Wort gemeint, sondern eine Person, die zur semantischen Klasse „Milliardär“ gehört. Ziel der semantischen Volltextsuche ist es somit den Suchbegriffen semantische Klassen zuzuordnen, die aus einer Menge von Entitäten („Bill Gates“, „Larry Page“, etc.) bestehen. Der Vorteil dieser Zuordnung liegt darin, dass man die Milliardäre nicht namentlich kennen muss, um diese zu finden.
Am Lehrstuhl für Algorithmen und Datenstrukturen an der Technischen Fakultät der Albert-Ludwigs-Universität Freiburg wurde bereits „Broccoli“, eine Demo einer semantische Volltextsuchmaschine, ausgearbeitet, die auf der englischsprachigen

Wikipedia läuft[1]. Das Ziel von Niklas Meinzer und mir war es, dies auch für deutsche Zeitungsartikel zu ermöglichen, wobei sich vieles der vorhanden Suchmaschine nutzen ließ. Zum Testen unserer Suche verwendeten wir etwa 90 000 Artikel einer bekannten deutschen Zeitung.
In Abbildung 1.1 ein Beispiel unseres Ergebnisses:

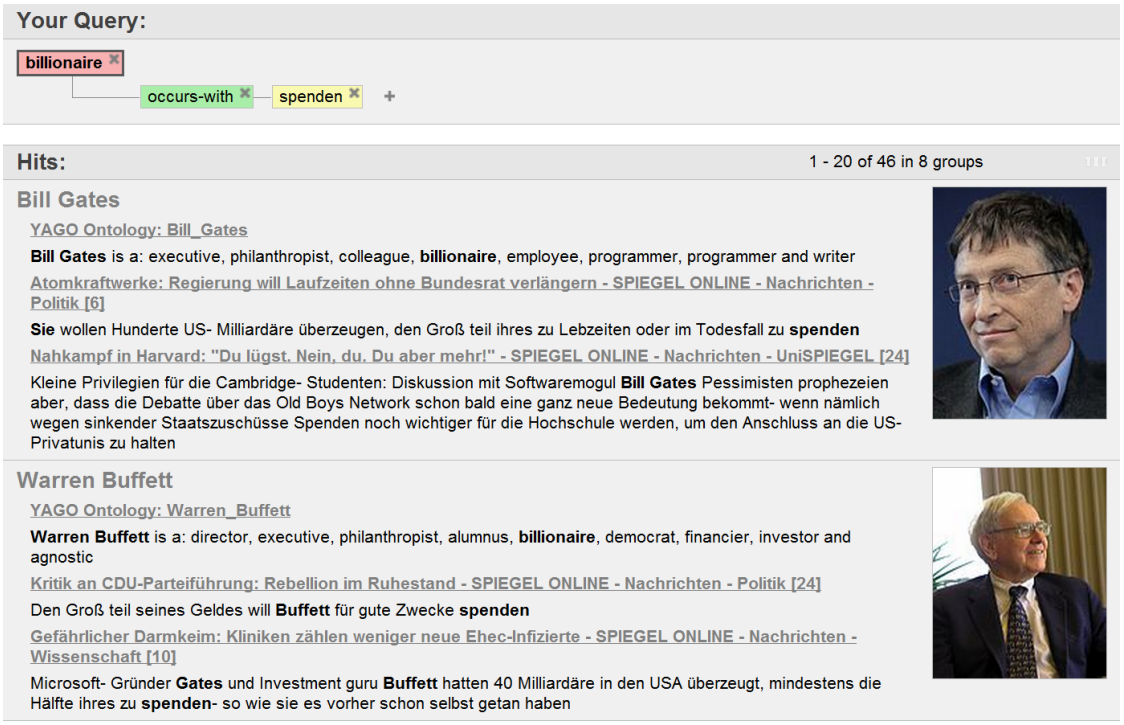

Abbildung 1.1: Ausschnitt aus einem Suchergebnis von BroccoliNewsSearch

Im oberen Bereich unter „Your Query“ wird unsere Suchanfrage ersichtlich: es soll in Artikeln nach Sätzen gesucht werden, die sowohl eine Entität aus der semantischen Klasse „Milliardär“ enthalten, als auch das Wort „spenden“.
Im unteren Bereich sieht man die Ergebnisse. Die Überschriften der einzelnen Suchergebnisse sind dabei Menschen, die zur semantischen Klasse „Milliardäre (billionaires)“ gehören. In den Auflistungen unter ihnen wird zum einen ersichtlich, zu welchen Klassen die einzelnen Milliardäre gehören („YAGO Ontology“) und zum anderen jene Sätze inklusive der dazugehörigen Zeitungsartikel, in denen diese Menschen in Zusammenhang mit Spenden erscheinen.

[1] http://stromboli.informatik.uni-freiburg.de:6222/Broccoli2/#

1.1 Semantische Volltextsuche

Wie genau semantische Volltextsuche funktioniert wird im folgenden Abschnitt erklärt. Die zugrundeliegenden Verfahren der Suche lassen sich dabei in drei wesentliche Schritte einteilen: Entitätenerkennung, Zuordnung der semantischen Klassen, semantische Interpretation von Sätzen. Diese Schritte werden in den Grundlagen erläutert, um anschließend auf die bisherigen Lösungsansätze und die für diese Bachelorarbeit auftretenden Probleme einzugehen. Zur Erklärung wird folgender fiktiver Text verwendet:

> *Bill Gates sprach gestern zu einem Interviewer der New York Times auf einer Bank im Central Park. Thema war die Spendenaktion, an welcher viele amerikanische Milliardäre teilnahmen. Gates war einer von ihnen. Er spendete knapp die Hälfte seines Vermögens.*

1.1.1 Grundlagen

In vielen Fällen können Begriffe verschiedene Bedeutungen haben. Diese Problematik soll an folgendem Beispiel erklärt werden: Ziel sei es alle Milliardäre zu finden, deren Geldinstitute / Banken in der Nähe des Central Parks liegen. Betrachtet man den fiktiven Text, stellt man fest, dass hier eine Falle auftreten kann. Hier wird „Bank“ im Zusammenhang mit dem Central Park erwähnt, allerdings bedeutet „Bank“ in diesem Fall Sitzgelegenheit. Ebenso ist unter anderem mit „Gates“ der Milliardär „Bill Gates“ gemeint und nicht der englische Popmusiker „Gareth Gates“. Um den Text korrekt analysieren zu können, ist es somit notwendig, den einzelnen Begriffen (z.B. „Gates“) ihre korrekte Bedeutung („Bill Gates“) zuzuordnen. Diesen Schritt nennt man Entitätenerkennung.
Weiterhin ist es erforderlich die gefundenen Entitäten semantischen Klassen zuzuweisen. Im Falle von „Bill Gates“ wäre dies unter anderem die Klasse der Milliardäre. Hier ist es wichtig, dass die korrekten Entitäten gefunden wurden. Denn wird eine Entität falsch erkannt, so kann die semantische Klasse nicht korrekt zugeordnet werden und das Suchergebnis wird verfälscht sein.
Man nehme nun wieder die Suche nach allen Milliardären, die Geld spenden. Analysiert die Suchmaschine den fiktiven Text, so stellt sie sich die Frage: „Wer spendete Geld?“. Der Computer besitzt kein grundlegendes Verständnis für die Semantik eines Satzes, im Gegensatz zum Menschen. Für ihn gibt es somit mehrere Möglichkeiten, wie „Bill Gates“, „Gates“, „Er“, der „Interviewer“ oder die „amerikanischen Milliardäre“. Ohne Sinnverständnis könnte allerdings auch z.B. mit „Central Park“ der Spender gemeint sein. Die Aufgabe der semantischen Volltextsuche liegt nun unter anderem darin herauszufinden, um wen es sich bei den Spendern handelt, darin, die Sätze semantisch zu interpretieren.

1.1.2 Bisherige Arbeit

Wie bereits erwähnt, wurde bereits die semantische Volltextsuche Broccoli programmiert. Für diese werden die Titel der Artikel der englischsprachigen Wikipedia als mögliche Entitäten genutzt. Für die Entitätenerkennung können zusätzlich die Eigenschaften der Wikipedia, wie z.B. Verlinkungen und Weiterleitungen, genutzt werden. Ein Vorteil der Wikipedia liegt darin, dass viele Wörter innerhalb eines Artikels selbst Entitäten sind und auf die dazugehörige Seite verlinken. So würde man in unserem fiktiven Text unter anderem einen Link auf die „New York Times“ finden, auf den „Central Park“, aber auch auf die „Bank“ als Geldinstitut. Ist ein Wort nicht verlinkt, wird es in den meisten Fällen bereits weiter oben erwähnt und der Link kann übernommen werden. Ist dies nicht der Fall, so wird einfach die naheliegendste Entität genutzt, die sich zu dem unverlinkten Wort finden lässt.
Für die Auflösung der semantischen Klassen verwendet die bereits implementierte semantische Volltextsuche „YAGO“, eine Wissensbasis, die auf der englischen Wikipedia und WordNet aufbaut[2]. Mit Hilfe dieser Ontologie lassen sich Entitäten zu semantischen Klassen zuordnen, wie z.B. „Bill Gates“ zur Klasse „Milliardäre“, und Relationen festlegen, z.B. „Bill Gates ist Bürger von Amerika“ und „Angela Merkel ist Bürgerin von Deutschland“.
Für die semantische Interpretation von Sätzen lässt sich grundlegend folgende Methodik anwenden: „Er“ gehört zu den Anaphoren und stellt keine eigenständige Person dar, sondern bezieht sich auf den vorhergehenden Namen „Gates“. Damit wissen wir nun bereits, dass „Gates“ der Spender ist. Nun stellt sich weiterhin die Frage, um wen es sich bei „Gates“ handelt. Wirft man einen Blick in die Wikipedia[3], so stellt man fest, dass es hier verschiedene Möglichkeiten gibt. Die Auflösung dieser Mehrdeutigkeit ist in diesem Fall leicht, da „Bill Gates“ zu Beginn des fiktiven Textes vorkommt, und wurde an dieser Stelle bereits von der Entitätenerkennung vollzogen.

1.1.3 Problematik für deutsche Zeitungsartikel

In deutschen Zeitungsartikeln lässt sich der Vorteil der verlinkten Seiten nicht nutzen, da keine Verlinkungen vorhanden sind. Um dennoch Entitäten erkennen und ambige Entitäten auflösen zu können, war es im Rahmen dieser Bachelorarbeit notwendig, die Entitätenerkennung neu zu implementieren (siehe Entitätenerkennung). Eines der Ziele dieser Bachelorarbeit war es, möglichst viel von der bereits vorhandenen Implementierung zu nutzen. Unter anderem, wie semantische Klassen erkannt werden. Da die bisherige Suchmaschine auf YAGO aufbaut, welches nur auf Englisch vorhanden ist, war es nicht möglich, dieses direkt zu übernehmen. Um YAGO dennoch nutzen zu können, mussten die deutschen Entitäten auf die englischen Entitäten abgebildet werden (siehe Vorverarbeitung).

[2] http://www.mpi-inf.mpg.de/yago-naga/yago/
[3] http://de.wikipedia.org/wiki/Gates

Weiterhin musste die semantische Interpretation der Sätze aus dem Englischen ins Deutsche geändert werden. Dies war allerdings nicht Bestandteil der Bachelorarbeit und wurde von Björn Buchhold implementiert.

1.2 Zielsetzung

Ziel dieser Arbeit ist es eine für deutsche Zeitungsartikel funktionierende semantische Volltextsuche vorzustellen. Dafür wurde ein möglichst großer Teil der bereits bestehenden Suchmaschine genutzt, sodass ausschließlich die Entitätenerkennung neu implementiert werden musste. Um eine funktionierende Entitätenerkennung durchzuführen, war es notwendig, Vorwissen zu definieren und zu speichern. Vorwissen, welches das Auffinden von Entitäten, das Mapping auf die englischen Entitäten und das Auflösen ambiger Entitäten ermöglicht. Ziel der Entitätenerkennung war es anschließend, mit Hilfe des gegebenen Vorwissens, Entitäten innerhalb eines Textes zu finden und dann zu entscheiden, welche der möglichen Entitäten jene mit der zutreffendsten Bedeutung sind.

1.3 Beitrag der Arbeit

Aufbauend auf der semantischen Volltextsuche Broccoli, wird in dieser Arbeit einer Erweiterung vorgestellt, die es ermöglicht die Suchmaschine auf beliebige deutsche Texte auszuweiten. Probleme kann es mit fachspezifischen Texten geben, da dahingehend nicht genügend Entitäten vorhanden sein könnten. Die Suchmaschine sollte auf ähnlich aufgebauten Sprachen agieren können. Dies wurde allerdings nicht getestet. Um dies bewerkstelligen zu können, müsste das Vorwissen (siehe Vorverarbeitung) neu generiert werden. Weiterhin müssten das Stemming, ein Algorithmus, um Wörter auf ihre Stammform zurückzuführen, und die Kompositazerlegung, welches das Auffinden von Entitäten in zusammengesetzten Wörtern ermöglicht, in Bezug auf die Sprache angepasst werden.
Weiterhin bietet diese Arbeit eine Methode, leicht und sicher Wörter in Texten auszusortieren, die als Entitäten nicht geeignet sind. Dies geschieht mit Hilfe eines PosTaggers, der den Wörtern eines Artikels die dazugehörige Wortart zuweist, und einer Liste unwichtiger Wörter. Hierdurch sinkt die Wahrscheinlichkeit, Entitäten für ungeeignete Wörter, wie z.B. Verben und Pronomen, zu finden, auf ein Minimum.
Da ein Mapping der deutschen Entitäten auf die englischen Entitäten notwendig ist, ist die Anzahl der zu findenden Entitäten beschränkt. Jedoch stellt das Auslesen der Weiterleitungsseiten (Meinzer, 2011) und der Disambiguierungsseiten der Wikipedia sicher, dass möglichst wenige Entitäten fehlen, sodass ein Großteil der Entitäten gefunden werden kann.
Je mehr Entitäten korrekt erkannt werden, umso besser sind die anschließenden

Suchergebnisse. Einen wesentlichen Beitrag dazu leistet das Scoring, dessen Ziel es ist, Wörtern die treffendsten Entitäten zuzuordnen. Durch eine zusätzliche Kontextkomponente bleibt das Scoring möglichst dynamisch und ist dazu in der Lage in verschiedenen Texten dem gleichen Wort unterschiedliche Bedeutungen zuordnen zu können. Das Scoring ist simpel gehalten und leicht erweiterbar, sodass es in Zukunft ohne Probleme an neue Ideen angepasst werden kann. Zusätzlich wurden Stemming (Meinzer, 2011) und eine Kompositazerlegung implementiert, durch welche mehr Entitäten gefunden werden können. Dies bietet den Vorteil, dass auch leicht abgeänderte Begriffe zu Suchergebnissen führen.
Weiterhin wurde eine Evaluation implementiert, die sowohl graphisch als auch prozentual einen Überblick über die Güte der Entitätenerkennung bietet. Diese läuft auf Basis von Ground Truths, welches Dateien sind, die eine perfekte Entitätenerkennung repräsentieren und mit denen die tatsächliche Entitätenerkennung verglichen wird. Zusätzlich ermöglicht es die Evaluation, leicht Ground Truths zu erstellen, der dann lediglich manuell die korrekten Entitäten hinzugefügt werden müssen. Um bereits einen Ansatz zu bieten, wurden mehrere Ground Truths erstellt und korrigiert. Um die Übersichtlichkeit zu erhöhen und den Code leicht verständlich zu machen, gibt es natürlich eine Dokumentation und eine Log-Ausgabe in verschiedenen Detailstufen.

2 Vorverarbeitung

Im folgenden Abschnitt wird auf die notwendige Vorverarbeitung vor der Entitätenerkennung eingegangen. Um Entitäten korrekt erkennen zu können, benötigen wir Vorwissen und Artikel, auf denen wir agieren können. Zu dem Vorwissen gehören unter anderem: „Welche Entitäten existieren?“, „Wie lassen sie sich Wörtern zuordnen?“. Wie wir mit Hilfe der Wikipedia zu diesem Wissen gelangen und wie wir die deutschen Entitäten auf die englischen Entitäten abbilden, wird im ersten Abschnitt erklärt. Anschließend wird auf das Erlangen der Zeitungsartikel eingangen und wie sie möglichst vorteilhaft gespeichert werden können. Zusätzlich wird erläutert, wie mithilfe der Artikel zusätzliches Wissen gewonnen werden kann, welches die Entitätenerkennung verbessert. Im letzten Abschnitt wird schließlich erklärt, wie unter Verwendung von YAGO Kontextwissen gewonnen werden kann.

2.1 Parsen der deutschen Wikipedia

Da die englische semantische Volltextsuche auf dem englischsprachigen Wikipedia aufbaut, ist es sinnvoll, zum Auffinden der deutschen Entitäten, auf die deutschsprachige Wikipedia zurückzugreifen.
Die Wikipedia stellt eine XML-Datei mit allen Artikeln in ihrer Datenbank bereit, die den Vorteil bietet, dass sich Informationen, wie z.B. Titel, Texte und Übersetzungen, leicht mit Hilfe von Tags extrahieren lassen. Diesen Vorteil nutzen wir, um die Menge der möglichen Entitäten zu erhalten. Dabei nutzen wir die Titel der einzelnen Wikipedia-Seiten als Entitäten. Wie bereits erwähnt, ist es für die Zusammenführung von Broccoli auf die von uns programmierte Suchmaschine notwendig, die deutschen Entitäten auf die englischen Entitäten abzubilden. Um dies umzusetzen, werden beim Parsen lediglich jene deutschen Wikipedia-Seiten genutzt, für die eine Übersetzung ins Englische vorhanden ist. Wir speichern diese Entitäten in einer Liste mit zwei Spalten. In der Ersten stehen die deutschen Entitäten, und in der zweiten Spalte deren englisches Pendant (Meinzer, 2011).
In den folgenden Unterabschnitten wird nun erwähnt, wie weitere Entitäten gewonnen werden können.

2.1.1 Weiterleitungen

Neben den bekannten Artikelseiten stellt die Wikipedia auch Weiterleitungen bereit. So wird z.B. „Angela Dorothea Merkel“[1] auf „Angela Merkel“[2] weitergeleitet. Weiterleitungsseiten besitzen grundsätzlich keine Übersetzungen, wodurch diese nicht in die Entitätenliste aufgenommen werden. Aus diesem Grund extrahieren wir auch diese Seiten aus der Wikipedia und speichern sie in einer eigenen Liste, welche ebenso wie die Entitätenliste zwei Spalten besitzt: den Artikelnamen der Weiterleitung und den Titel des Ziels (Meinzer, 2011).

2.1.2 Disambiguierungsseiten

Neben den Artikeln und den Weiterleitungen gibt es auf Wikipedia zusätzlich sogenannte Disambiguierungsseiten oder auch Begriffsklärungsseiten, wie z.B. „Netz“[3]. Diese Seiten listen mögliche Bedeutungen des Wortes auf. So könnte „Netz“ für „Stromnetz“ wie auch für „World Wide Web“ stehen. Diese Seiten sind sehr wichtig zur Erkennung der korrekten Bedeutung eines Begriffs. Auch diese werden tabellarisch in einer Liste gespeichert. Jeder Disambiguierungsseite wird dabei in der gleichen Reihenfolge die Menge ihrer möglichen Bedeutungen zugewiesen.

2.2 Parsen der Artikel

Die von uns benutzen Artikel stammen von einer bekannten deutschen Zeitung und werden mithilfe eines Crawlers heruntergeladen und anschließend in einer CSV-Datei mit folgendem Aufbau gespeichert: Dokumentennummer, Datum, URL, Titel, Artikel (Meinzer, 2011). Dieses Format bietet den Vorteil möglichst viele Informationen speichern und die gewünschte Information schnell auslesen zu können.
Folgend werden nun weitere Methoden der Vorverarbeitung erläutert, die, aufbauend auf den Artikeln, Optimierungen für die Entitätenerkennung ermöglichen.

2.2.1 Part-Of-Speech Tagging

In vielen Fällen erweist sich das Erkennen von Entitäten als problematisch, wenn man den Kontext nicht kennt. So wird z.B. „wirft“ als die Ortsgemeinde „Wirft“[4] erkannt. Kennt man die Wortart eines Begriffes, so lassen sich viele Wörter bereits ausschließen. Dies lässt sich mithilfe eines Part-Of-Speech Taggers oder kurz PosTagger bewerkstelligen. Wir nutzen dabei einen PosTagger der Universität Stuttgart,

[1] http://de.wikipedia.org/wiki/Angela Dorothea Merkel
[2] http://de.wikipedia.org/wiki/Angela Merkel
[3] http://de.wikipedia.org/wiki/Netz
[4] http://de.wikipedia.org/wiki/Wirft

den sogenannten TreeTagger (Schmid, 1994). Dieser bestimmt mit einer Genauigkeit von über 96 % die korrekte Wortart eines Wortes (Schmid, 1995). Mithilfe dieses PosTaggers würde im gegebenen Fall festgestellt werden, dass das Wort zur Kategorie Verb gehört und dementsprechend als Entität auszuschließen ist. Weiterhin hilft das PosTagging zusammengehörige Namen zu bestimmen. So kann man davon ausgehen, dass, wenn zweimal ein Eigenname auftaucht (z.B. „Angela Merkel“), es sich dabei um einen zusammengehörigen Namen handelt. Ohne dieses Feature können Namen fehlerhaft erkannt werden, falls der gesamte Name keine Entität ist (z.B. „Michael“ in „Michael Klammhausen“ als „Michael Ballack“). Aus diesem Grund bearbeiten wir die Artikel der CSV ein weiteres Mal und posTaggen diese. Die einzelnen Wörter haben anschließend folgendes Format: $<$*Wort*$>$ $<$*Wortart*$>$, so dass der Text nach dem PosTagging ungefähr wie folgt aussieht: *Die#Artikel Kommune#Nomen von#Präposition Michael#Eigenname ist#Verb arm#Adjektiv.*[5]

2.2.2 Unwichtige Wörter

In wenigen Fällen schlägt die Erkennung des PosTaggers fehl oder ist für die Aufgabe nicht geeignet, wie zum Beispiel bei der Zerlegung zusammengesetzter Substantive. Um diesen Fehlern begegnen zu können, wurde zusätzlich eine Liste unwichtiger Wörter und Füllwörter generiert. In dieser Liste stehen inbesondere Wörter, die keinesfalls Entitäten sein können: Artikel, Füllwörter, Konjunktionen, Präpositionen, Pronomen, etc.. Diese erhalten wir durch das Parsen aller Artikel und des Herausfilterns aller Wörter, die manchmal groß und manchmal klein geschrieben werden. Dies ist ein Vorteil der deutschen Sprache, sodass diese Methode in sehr vielen Fällen funktioniert. Um die Liste zu optimieren, wurde sie anschließend manuell nachbearbeitet, indem fehlerhaft erkannte Wörter entfernt und fehlende Wörter hinzugefügt wurden.

2.2.3 Zählen der Entitäten

In sehr vielen Fällen gibt es mehrere mögliche Entitäten zu einem Wort. Man nehme als Beispiel das Wort „Werk“[6]. Ohne ausreichendes Wissen über den Kontext des Artikels ist es nahezu unmöglich die korrekte Bedeutung dieses Wortes zu bestimmen. Wir gehen von der Annahme aus, dass in den meisten Fällen Entitäten vollständig benannt werden. Dabei definieren wir jene Entitäten als vollständig, deren gesamte Beschreibung im Artikel vorkommt. So ist „Angela Merkel“ vollständig, wenn „Angela Merkel“ im Artikel auftaucht und unvollständig, wenn lediglich „Merkel“ oder „Angela“ vorkommt. Existieren also für einen Begriff mehrere Bedeutungen, bekommen, ausgehend von unserer Annahme, jene Entitäten einen Bonus, welche häufig

[5] die Wortarten werden im PosTagger durch Kurzformen repräsentiert, wurden jedoch zur Veranschaulichung hier ausgeschrieben

[6] http://de.wikipedia.org/wiki/Werk

vollständig auftauchen. Aus diesem Grund parsen wir alle Artikel vor der eigentlichen Entitätenerkennung und zählen die vorkommenden, vollständigen Entitäten. Als vollständig bezeichnen wir dabei auch Entitäten, bei welchen eine Erklärung im Titel auftaucht, wie z.B. „Werk (Urheberrecht)“[7], wie auch alle Entitäten einer Disambiguierungsliste, falls ein ambiges Wort wie „Werk“ geparst wird. Aus diesem Verfahren resultiert eine Menge der naheliegstendsten Entitäten, welche inklusive der Anzahl ihrer Vorkommen in einer Liste gespeichert werden.

2.3 Parsen der YAGO-Facts

Wie in der Einleitung bereits erwähnt, arbeitet die semantische Volltextsuche mithilfe der Ontologie YAGO. Aus dieser Ontologie wurden bereits im Rahmen der bereits implementierten semantischen Volltextsuche am Lehrstuhl für Algorithmen und Datenstrukturen sogenannte YAGO-Facts abgeleitet (Buchhold, 2010, S. 30 ff). Die Menge dieser YAGO-Facts bauen einen gerichteten, azyklischen Graphen auf, welches später zur Kontexterkennung dienen sollen.
Wie das Graph genau aufgebaut ist und wie es zur Kontexterkennung genutzt werden kann, wird im Folgenden erklärt.
Zur Veranschaulichung des Graphen sieht man in Abbildung 2.1 einen Ausschnitt, in welchem die Pfade für „Angela Merkel“ hervorgehoben wurden.
Die Knoten lassen sich sowohl als semantische Klassen, als auch als Entitäten betrachten. So gehört „Angela Merkel“ zur Klasse „woman“, „person“, aber auch zur Klasse „object“, etc.. Die Entität „object“ gehört schließlich zu den semantischen Klassen „physicalentity“ und „entity“. Der Wurzelknoten ist damit die semantische Klasse aller Entitäten und somit die abstrakteste Klasse innerhalb des Netzes, wohingegen die Blätter, wie „Angela Merkel“, nur noch Entitäten und keine weitere Klasse darstellen. Diese Fakten „*entity* ist Klasse von *physicalentity* ist Klasse von ... ist Klasse von *Angela Merkel*“ stellen nun die YAGO-Facts dar.
Die YAGO-Facts für „Angela Merkel“ sehen damit wie folgt aus:

entity:physicalentity:object:whole:livingthing:organism:person:leader:head
entity:physicalentity:object:whole:livingthing:organism:person:leader:politician
entity:physicalentity:object:whole:livingthing:organism:person:adult:woman
entity:physicalentity:object:whole:livingthing:organism:person:female:woman

Unter Verwendung dieser YAGO-Facts lässt sich nun der Kontext einer Entität bestimmen. So gehören sowohl „Angela Merkel“ als auch „Helmut Kohl“ zur Klasse „Personen“. Damit lässt sich der Graph auch innerhalb der Entitätenerkennung nutzen. Da wir Artikel einer bekannten deutschen Zeitung nutzen, kann man davon ausgehen, dass jeder Artikel einem bestimmten Themenbereich angehört. Aus diesem Grund sollten viele Entitäten innerhalb des Artikels gleiche Vorfahren besitzen.

[7] http://de.wikipedia.org/wiki/Werk (Urheberrecht)

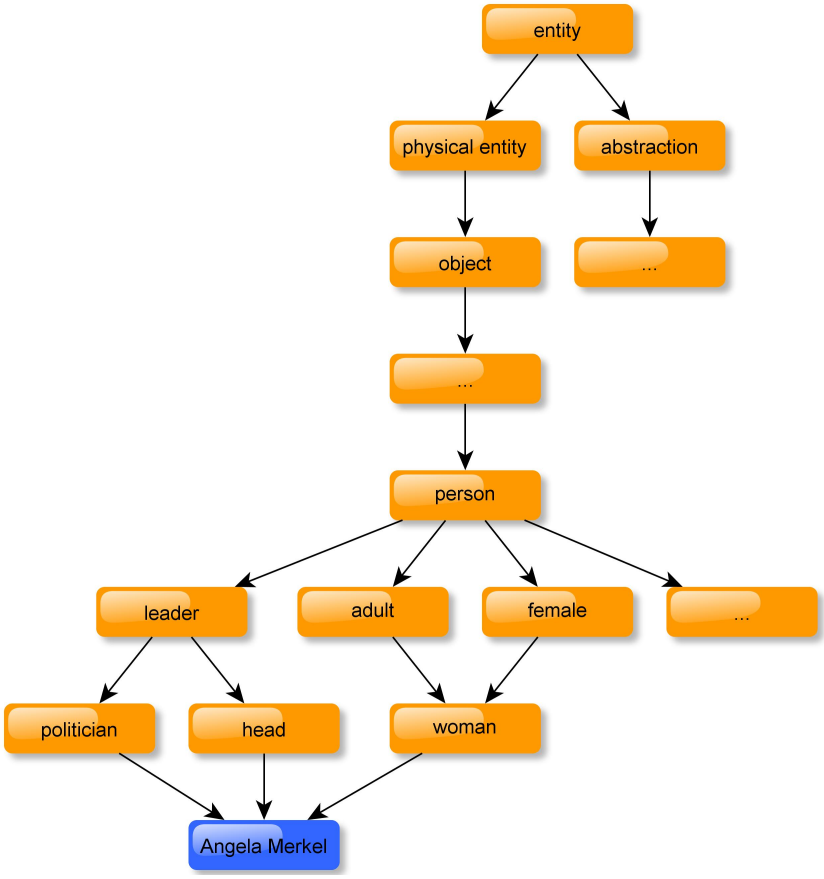

Abbildung 2.1: Ausschnitt des Graphen

Als Beispiel: wurde bereits „Angela Merkel“ gefunden und im späteres Verlauf wird „Kohl“ erwähnt, so kann man mit Hilfe der YAGO-Facts darauf schließen, dass „Helmut Kohl“ und nicht das Gemüse „Kohl“ gemeint ist. Mehr dazu unter Scoring
Aus diesem Grund parsen wir diese YAGO-Facts und speichern zu jeder Entität die dazugehörigen Pfade in einer Liste.

3 Methoden der Entitätenerkennung

Im Folgenden werden einige Methoden erklärt, um das spätere Verständnis der Entitätenerkennung zu verbessern. Die Methoden werden angewendet, um die Ergebnisse der Entitätenerkennung zu optimieren.

3.1 PosTagging

Wie im Kapitel Vorverarbeitung bereits erwähnt, werden dem Artikel PosTags hinzugefügt, die die Wortarten der einzelnen Begriffe repräsentieren. Diese werden beim Parsen des Artikels wieder entfernt und, inklusive ihrer Position, gespeichert, um sie in der Entitätenerkennung den Wörtern zuweisen zu können.
Sie werden anschließend verwendet um Begriffe, die als Pronomen, Artikel, Verben, etc. erkannt werden, als Entität in der Entitätenerkennung ausschließen zu können.

3.2 Stemming

Unter Stemming versteht man das Zurückführen eines Wortes auf seine Stammform. Zum Beispiel ist die Stammform von „Ehrzeiges“ „Ehrgeiz“. In geschriebenen Texten und Artikeln trifft man sehr häufig morphologische Varianten eines Wortes. Stemming ist ein Ansatz, um die dazugehörigen Entitäten dennoch erkennen zu können. Ziel des Algorithmus ist es, typische Endungen der deutschen Sprache von den Worten zu entfernen. Typische Endungen sind dabei „e“, „s“, „n“, „t“, „em“, „er“ und „nd“.
Bei der Entfernung dieser Endungen geht der Algorithmus, je nach Endung, von einer Mindestlänge des Wortes aus. So muss z.B. gelten, dass nd nur entfernt werden darf, wenn das Wort mindestens 5 Zeichen lang ist. Ebenso sollte „t“ nicht von Substantiven entfernt werden, da dies eine untypische Endung für Nomen ist. Dieser von Meinzer (2011) implementierter Algorithmus kann nun je nach Bedarf rekursiv angewandt werden, so würde im Falle „Ehrgeizes“ zuerst das „s“ und anschließend das „e“ entfernt werden.

3.3 Kompositazerlegung

Komposita sind zusammengesetzte Worte, wie „Umzugskarton“. Ziel der Zerlegung ist es die Worte „Umzug“ und „karton“ zu erhalten.
Die Kompositazerlegung baut auf ähnlichen Prinzipien wie das Stemming auf. In der deutschen Sprache gibt es typische Fugenlaute, die zwei Wörter voneinander trennen: „e“, „s“, „es“, „n“, „en“, „er“, „ens“[1].
Im Algorithmus wird nun an jeder Stelle versucht das Wort zu teilen und, falls möglich, die Buchstabenfolgen zu entfernen. Es wird auch hier von einer Mindestlänge des Wortes ausgegangen, sodass gilt, dass die Fugenlaute frühstens nach den ersten drei und vor den letzten drei Buchstaben auftreten dürfen.
Alle möglichen Zerlegungen werden durchgegangen und es wird geprüft, ob es sich bei beiden Worten um Entitäten handelt. Ohne diese Bedingung können Zerlegungen wie „Date-nstrukturen“ gefunden werden, die keinerlei Sinn ergeben. Für alle Zerlegungen, für die diese Bedingung erfüllt ist, wird ein Scoring angewendet und die Wörter des besten Paares werden der Teilentitätenerkennung übergeben.

[1] http://de.wikipedia.org/wiki/Fugenlaut

4 Entitätenerkennung

In diesem Kapitel beschäftigen wir uns mit der Entitätenerkennung. Diese vollzieht sich in 2 größeren Schritten: die Erkennung eindeutiger Entitäten und die Erkennung uneindeutiger Entitäten. Die Erkennung eindeutiger Entitäten werden wir im Folgenden auch Multientitätenerkennung und die der uneindeutigen Entitäten Teilentitätenerkennung nennen. Dies begründet sich darauf, dass in der Erkennung eindeutiger Entitäten, fast nur Entitäten gefunden werden, die aus mehr als einem Wort bestehen (z.B. „Angela Merkel“). In der Erkennung uneindeutiger Entitäten finden wir hingegen oft Entitäten, die nur zum Teil enthalten sind (z.B. „Angela“ für „Angela Merkel“.
Zusätzlich wird auf die verwendete Datenstruktur eingegangen. Im Folgenden ein Überblick, in welchen Schritten die Entitätenerkennung abläuft:

Algorithmus 1 Entitätenerkennung

Eingabe: CSV und Listen aus der Vorverarbeitung
Ausgabe: alle Artikel mit den gefundenen Entitäten im Wikipediaformat

```
1: readInLists(listsOfPreprocessing)
2: for all articles do
3:     storeAndRemovePosTags(article)
4:     multiEntityRecognition()
5:     partWordEntityRecognition(ambiguousEntities)
6:     storeInWikipediaFormat(foundEntities)
7: end for
```

Wie in der Vorverarbeitung bereits erwähnt nutzen wir eine CSV-Datei, um die Artikel zu speichern zu können. Weiterhin verwenden wir Listen, die uns das notwendige Vorwissen bereitstellen, um Entitäten korrekt erkennen zu können. In der Entitätenerkennung lesen wir nun zuerst die Listen aus der Vorverarbeitung ein (Zeile 1) und entfernen aus dem Artikel die PosTags (Zeile 2). Anschließend wenden wir auf jeden Artikel die Entitätenerkennung (Zeile 4, 5) an. Die gefundenen Entitäten werden danach in Zeile 5 in einem, dem Wikipedia ähnlichen, Format gespeichert (Meinzer, 2011), welche anschließend von der bereits am Lehrstuhl für Algorithmen und Datenstrukturen implementierten Suchmaschine genutzt werden kann.

4.1 Entitäten

Um besser auf Entitäten agieren zu können, wird ein eigenes *Struct* genutzt. Dieses enthält im wesentlichen den Namen der deutschen Entität, den Namen der englischen Entität, ein Feld für die Menge ihrer Disambiguierungen und eines für die Menge derjenigen Entitäten, von welcher sie ein Präfix ist. Hier ein Beispiel für die Entität „Angela“[1]:

- deutscher Name: Angela
- englischer Name: Angela (given name)
- Disambiguierungen: leer, da es sich um keine Disambiguierungsseite handelt
- Präfix von: Angela Merkel, Angela Stresemann, Angela Stachowa, usw.

Diese Entitäten erhalten wir durch Einlesen der in der Vorverarbeitung genannten Listen. Zusätzlich haben wir uns entschieden, alle Entitäten auch rückwärts („Angela Merkel ⇒ Merkel Angela“) zu speichern. Der Grund hierfür wird im späteren Verlauf deutlich.

4.2 Datenstruktur

Um auf einer großen Datenmenge agieren zu können, ist es notwendig, die Entitätenerkennung möglichst effizient zu gestalten. Das wichtigste Element ist bei diesem Datenumfang eine effiziente Datenstruktur, die es erlaubt schnell und möglichst platzsparend Entitäten zu finden. Um dies zu bewerkstelligen wird ein sogenannter Trie benutzt, ein für Zeichenketten modifizierter Baum. Der wesentliche Vorteil dieser Datenstruktur liegt darin, dass in $\Theta(n)$ festgestellt werden kann, ob ein Wort der Länge n in diesem Baum enthalten ist. Selbiges gilt für das Einfügen einer Zeichenkette. Ebenso ist der Trie sehr platzeffizient. Im folgenden Abschnitt soll nun der genaue Aufbau eines Tries erklärt werden und erläutert werden, wie die genannte Effizienz zustande kommt.

4.2.1 Aufbau

Der Aufbau des von uns genutzten Trie wird in Abbildung 4.1 ersichtlich.
Zum besseren Verständnis der Datenstruktur hier ein Beispiel, das Wort „Haus“. Dieses Wort wird dem Trie übergeben, der anschließend nach jedem Buchstaben den Baum hinabwandert. Beginnend beim Wurzelknoten wird der Buchstabe „H“ einer Map übergeben, welche anschließend in $\Theta(1)$ den Zielknoten „H“ zurückgibt. Anschließend wird der Map der Buchstabe „a“ übergeben, woraufhin der Zielknoten

[1] http://de.wikipedia.org/wiki/Angela

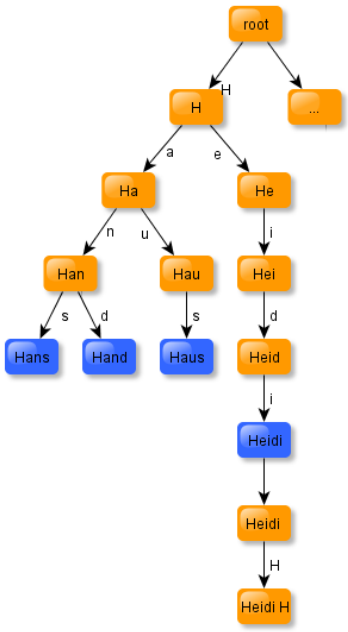

Abbildung 4.1: Ausschnitt eines Tries

„Ha“ zurückgeben wird. Folgend die Buchstaben „u“ und „s“. Gelangt der Algorithmus an eine Stelle für den kein Eintrag für den Folgebuchstaben vorhanden ist, so bedeutet dies, dass das Wort im Trie nicht enthalten ist. Ansonsten gibt der Algorithmus jenen Knoten zurück, der entweder ein Blatt ist oder für den mithilfe eines Boolean gilt, dass der Knoten ein Ende einer Entität ist. Diese sind in Abbildung 4.1 blau markiert. Anstelle von einfachen Wörtern, speichern wir innerhalb des Tries die Entitäten, wobei der Trie entlang des deutschen Namens einfügt und sicht.
Da die Map in $\Theta(1)$ Buchstaben einfügen und auslesen kann, liegt die Laufzeit, wie oben bereits erwähnt, für ein Wort der Länge n in $\Theta(n)$. Weiterhin belegen viele Wörter denselben Platz. So haben „Haut“ und „Haus“ den gemeinsamen Präfix „Hau“. Dadurch wird für diese beiden Worte lediglich ein Platz für 5 Buchstaben benötigt, anstelle von 8. Bei dem von uns verwendeten Datenumfang ist dies eine sehr gute Möglichkeit, die Entitäten platzsparend zu speichern.

4.2.2 Modifikationen

In vielen Fällen passen Entitäten und die zu suchenden Begriffe nicht perfekt übereinander. Zum Beispiel könnte die Entität „Email“ als „e-mail“ im Artikel enthalten sein. Gründe für die Verschiedenheit sind neben anderen Schreibweisen, die oft durch Weiterleitungen abgefangen werden, in den meisten Fällen Unterschiede in der Groß- und Kleinschreibung, sowie in der Zeichensetzung.

Um diesen Problemen begegnen zu können, werden, vor der Suche innerhalb des Tries, alle Großbuchstaben einer Zeichenkette in kleine Buchstaben umgewandelt (Meinzer, 2011). Zusätzlich werden Bindestriche als Leerzeichen betrachtet.

4.3 Regeln

Um die korrekten Entitäten zu einer Phrase oder einem Wort zu finden, kann man verschiedene, meist geltende Regeln definieren und anwenden. Diese werden im Folgenden in Reihenfolge ihrer Bedeutung für die Entitätenerkennung definiert. Stemming und das Ausschließen unwichtiger Wörter werden in der gesamte Entitätenerkennung genutzt. Die Vollständigkeits- und Kompositaregel werden in der Multientitätenerkennung angewendet, wohingegen die weiteren Regeln in der Teilentitätenerkennung genutzt werden.

Unwichtige Wörter

Unwichtige Wörter sind oft Teile von Entitäten. So kommt das Wort „Und“ zu Beginn zahlreicher Entitäten vor. Wir gehen von der Annahme aus, dass, wenn ein unwichtiges Wort zu Beginn oder auch am Ende einer Entität vorkommt, die Entität vollständig repräsentiert sein muss, da es sich in solchen Fällen meist um Buchtitel, Filmtitel, Serien, etc. handelt.
Weiterhin gilt, dass eine Entität, die eine Länge von einem Wort hat, nicht unwichtig sein darf. Unwichtig bedeutet in diesem Fall, dass es kein Nomen ist.

Vollständige, eindeutige Entitäten

Wenn eine Entität, wie „Angela Merkel“, innerhalb eines Textes auftaucht, so kann man davon ausgehen, dass hiermit eine vollständige, eventuell sogar eindeutige Entität gemeint ist und nicht „Angela“ als Vorname und „Merkel“ als texanische Stadt.

Gefundene Entitäten

In vielen Fällen wiederholen sich Entitäten innerhalb von Texten. Wurde z.B. einmal „Angela Merkel“ genannt, so ist die Wahrscheinlichkeit sehr hoch, dass beim Auftauchen von „Merkel“ wieder „Angela Merkel“ und nicht „Merkel als Stadt von Texas“ gemeint ist.

Simpelste Entität

Entitäten werden selten abgekürzt und sind meist vollständig in Texten enthalten. Es wird also von der Annahme ausgegangen, dass die simpelste Entität die wahrscheinlichste Entität ist. Ein Beispiel: taucht nur das Wort „Partei“ im Text auf, ist

die Wahrscheinlichkeit sehr gering, dass damit die „Partei Bibeltreuer Christen“ gemeint ist. Viel mehr ist davon auszugehen, dass die „Partei“ im Allgemeinen gemeint ist.

Disambiguitäten

Für viele Disambiguierungsseiten gilt, dass die häufigste Bedeutung an erster Stelle steht und die Wahrscheinlichkeit für die richtige Bedeutung mit jedem Platz weiter unten in der Liste sinkt. Dieser Vorteil lässt sich in der Entitätenerkennung nutzen, sodass man, innerhalb des Scorings, Entitäten, die weit oben in der Liste stehen, einen Bonus gibt.
Weiterhin ist davon auszugehen, dass, wenn die simpelste Entität eine Disambiguierungsseite ist, eine von ihren Bedeutungen treffend ist und die folgenden Präfix-von-Seiten gar nicht weiter beachtet werden müssen.

Stemming

In vielen Fällen finden wir Wörter, die nicht ihrer Stammform entsprechen und daher nicht gefunden werden können. Insofern es also möglich ist, sollte ein Wort auf seine Stammform gebracht und ebenso die daraus entstehenden Entitäten betrachtet werden.

4.4 Multientitätenerkennung

Ziel der Multientitätenerkennung ist es eindeutige Entitäten zu finden. Unter eindeutigen Entitäten verstehen wir Entitäten, deren Präfix-von-Liste leer ist. Denn in diesem Fall gibt es nur eine mögliche Bedeutung eines Wortes, bzw. einer Phrase. Weiterhin sortiert die Multientitätenerkennung jene Phrasen und Wörter aus, die mehrdeutig sind und gibt sie an die Teilentitätenerkennung weiter.
Im Folgenden ein Beispiel zur Veranschaulichung:

> *Gestern wurde Angela von Fliegen angegriffen, während Christian Wulff auf der Sessellehne saß.*

In diesem Beispiel wird „Christian Wulff“ als eindeutige Entität erkannt werden. „Gestern“, „während“, „Sessellehne“, „saß“, „Fliegen“ werden als mehrdeutige Entitäten erkannt werden, da ihre Präfix-von-Liste nicht leer ist, und anschließend an die Teilentitätenerkennung weitergegeben. Selbiges gilt für die Phrase „Angela von“, welche z.B. für die Entität „Angela von Foligno“ stehen könnte, wie auch für „auf der“, welche Präfix zahlreicher Film-, Buch- und Serientitel ist.

Algorithmus 2 Multientitätenerkennung

Eingabe: einen Artikel aus der CSV
Ausgabe: eindeutige Entitäten, uneindeutige Entitäten

```
1:  for all words in article do
2:    phrase ← word
3:    while isEntity(phrase) or hasPrefixes(entity) do
4:      if isEntity(phrase) and not hasPrefixes(entity) then
5:        add entity to found entities
6:        break and go on in for loop()
7:      end if
8:      phrase + = nextWord
9:    end while
10:   if isEntity(phrase) then
11:     add entity to ambiguous entities
12:   else if stemmedWordIsEntity(word) then
13:     add entity to ambiguous entities
14:   else if isKomposita(word) then
15:     add entities of word to ambiguous entities
16:   end if
17: end for
```

Die Erkennung läuft mit Hilfe einer Schleife über alle Wörter innerhalb eines Artikels. Das Auslesen der einzelnen Worte wurde mit einem leicht modifizierten, effizienten Parser bewerkstelligt, welcher bereits im Rahmen der Suchmaschinenprogrammierung am Lehrstuhl für Algorithmen und Datenstrukturen geschrieben wurde.

Wir wissen bereits aus dem Abschnitt Entitäten, dass jede Entität eine Präfix-von-Liste enthält. Zur Erinnerung enthält z.B. die Präfix-von-Liste der Entität „Angela“, unter anderem die Entitäten „Angela Merkel“,„Angela Stresemann“ und „Angela Stachowa“. Dies kann für die Multientitätenerkennung genutzt werden.
Es werden solange nacheinander Wörter ausgelesen bis die Präfix-von-Liste leer ist und die Entität somit eindeutig ist (Zeile 4). In den meisten Fällen kommt es nicht hierzu und die Multientitätenerkennung bricht ab. Es konnte also keine eindeutige Entität gefunden werden. Dies ist z.B. nach „Angela von“ der Fall. Sind die gefunden Wörter eine Entität oder Teil einer Entität, wie im eben genannten Beispiel, so wird diese zur Liste der uneindeutigen Entitäten hinzugefügt (Zeile 10).
In vielen Fällen wird bereits für das erste Wort keine Entität gefunden. An dieser Stelle werden nun zwei verschiedene Versuche gestartet, dennoch eine Entität zu erkennen: Stemming (Zeile 12) und Kompositazerlegung (Zeile 14). So wird das Stemmingverfahren auf das Wort „Fliegen“ angewendet und „Fliege“ zurückgegeben. „Sessellehne“ wird in der Kompositazerlegung in „Sessel“ und „Lehne“ aufgeteilt. Ebenso werden diese zwei Verfahren z.B. auf das Wort „wurde“ angewendet, allerdings ohne Erfolg.
Die Multientitätenerkennung wurde von Meinzer (2011) implementiert. Im Rahmen dieser Bachelorarbeit gab es zusätzliche Modifikationen. Dazu gehören die Kompositazerlegung, das Zuweisen der korrekten PosTags zu den Entitäten, sowie ein vorheriges Testen darauf, ob es sich bei einer gefundenen Entität, insofern sie nur ein Wort lang ist, um eine für Entitäten übliche Wortart handelt. Mehr dazu im Abschnitt PosTagging. Weiterhin wird eine erkannte Entität nach dem Stemming den uneindeutigen Entitäten hinzugefügt, anstatt, wie zuvor, den eindeutigen Entitäten.

4.5 Teilentitätenerkennung

Ziel der Teilentitätenerkennung ist es uneindeutigen Entitäten die korrekte Bedeutung zuzuordnen. Dazu wird ausschließlich auf Basis der von der Multientitätenerkennung erkannten, ambigen Entitäten agiert.
Dabei werden unwichtige Wörter aussortiert, die möglichen Bedeutungen (Entitäten) eines Wortes ermittelt und anschließend mit Hilfe eines Scoringverfahrens den Bedeutungen unterschiedliche Werte zugewiesen. Die Entität mit der besten Wertung wird anschließend den gefundenen Entitäten hinzugefügt.

4.5.1 Algorithmus

Algorithmus 3 Teilentitätenerkennung

Eingabe: uneindeutige Entitäten
Ausgabe: Entitäten, die der Liste der gefundenen Entitäten hinzugefügt werden

```
1:  for all ambiguous entities do
2:      erase stopwords
3:      if #words is 0 then
4:          continue;
5:      else if #words is 1 and not isEntityByPostag() then
6:          continue;
7:      end if
8:      if isPartOfEntity() then
9:          add entity to found entities
10:     else if #words > 1 then
11:         split into words and push each of them into the ambiguous vector
12:     end if
13: end for
```

Das Aussortieren der unwichtigen Wörter geschieht durch zwei Verfahren.
Beim Einlesen der Listen aus der Vorverarbeitung wurden alle unwichtigen Wörter, wie „und“, „obwohl“, in einem Trie gespeichert, sodass schnell erkannt werden kann, ob ein Wort unwichtig ist. So wird zuerst die uneindeutige Phrase auseinandergenommen und alle unwichtigen Wörter am Anfang und am Ende entfernt (Zeile 2). Ebenso werden Zahlen, die nicht mindestens vierstellig sind, gelöscht, da diese im Normalfall keine Entität darstellen. Längere, insbesondere vierstellige Zahlen, können hingegen wichtige Jahreszahlen sein. Nach dem Entfernen der unwichtigen Wörter würde die fiktive Phrase „Und die Partei von“ auf „Partei“ minimiert werden. Im dem Fall, dass es sich bei allen Wörtern der Phrase um unwichtige Wörter handelt, wird zur nächsten ambigen Entität übergegangen (Zeile 3).
Beim zweiten Verfahren wird in dem Fall, dass es sich um nur ein Wort handelt, betrachtet, ob die Wortart passend zu einer möglichen Entität ist. Ist die Wortart

weder Nomen noch Eigenname, so wird davon ausgegangen, dass es sich bei dem Wort um keine Entität handeln kann. Das Wort wird damit verworfen (Zeile 5).
Nach dieser Vorverarbeitung wird versucht eine passende Entität in dem Trie zu finden (Zeile 8). Im Trie werden nacheinander die Buchstaben der Phrase gesucht. Da der Trie nur vorwärts suchen kann, haben wir Entitäten, die aus mehreren Wörtern bestehen auch rückwärts gespeichert. So ist „Angela Merkel“ als „Angela Merkel“, aber auch als „Merkel Angela“ im Trie hinterlegt. Dadurch wird ermöglicht, dass auch die Teilentität „Merkel“ gefunden werden kann.
Wird keine passende Entität gefunden, und die Phrase besteht aus mehr als 2 Wörtern, so werden diese einzeln der ambigen Liste hinzugefügt, sodass sie später nochmals bearbeitet werden können (Zeile 10).
In der eigentlichen Entitätenerkennung wird nun versucht, die Menge aller möglichen Entitäten zu bekommen und darauf ein Scoring anzuwenden.

Algorithmus 4 isPartOfEntity

Eingabe: uneindeutige Entität
Ausgabe: gibt zurück, ob eine passende Entität existiert und falls ja welche

```
1:  get possible entities by using prefixOf list or disambiguation list
2:  get more possible entities by stemming up to 3 times
3:  for all possible entities do
4:     if alreadyFoundBefore(entity) then
5:        add to already found vector
6:     end if
7:  end for
8:  if #alreadyFoundVector > 1 then
9:     substitute with possible entities vector
10: else if #alreadyFoundVector is 1 then
11:    return entity in this vector
12: end if
13: filter possible entities
14: correctEntity ← doScoring(possible entities)
15: return correctEntity
```

Hierbei wird nun zuerst die Menge der möglichen Entitäten zusammengesucht. Dies geschieht indem, für die gegebene Phrase, die Entität innerhalb des Tries gesucht wird. Handelt es sich dabei um eine Disambiguierungsseite, so werden die möglichen Disambiguierungen als Liste der möglichen Entitäten behandelt. Handelt es sich um einen normalen Artikel, so werden der Liste, neben der gegebenen Entität, alle Entitäten hinzugefügt, von welcher sie Präfix ist (Zeile 1).
Anschließend wird versucht, insofern es sich nur um ein Wort handelt, das Wort auf einen Wortstamm zurückzuführen. Das geschieht in maximal 3 Iterationen, da Endungen üblicherweise nicht mehr als 3 Buchstaben lang sind. Alle möglichen Entitäten, die dabei gefunden werden, werden wiederum der Liste der möglichen En-

titäten hinzugefügt (Zeile 2).
Nun wird überprüft, ob eine Menge der möglichen Entitäten bereits in den gefundenen Entitäten zu finden ist (Zeile 3). Falls bereits eine dieser Entitäten gefunden wurde, so wird diese als die passende Entität gewählt (Zeile 10). Wurden sogar mehrere passende Entitäten entdeckt, so wird die Liste der möglichen Entitäten mit dieser Liste ersetzt.
Wie bereits unter Part-Of-Speech Tagging erwähnt, ist es notwendig zusammengehörende Namen wie „Michael Klammhausen“ zu erkennen. Aus diesem Grund betrachten wir hier ein weiteres Mal die Wortart des aktuellen Wortes. Ist dieses ein Eigenname, z.B. „Michael“ betrachten wir zusätzlich die Wortart des vorhergehenden und folgenden Wortes im Text. Nehmen wir an, das folgende Wort sei ein Eigenname, z.B. „Klammhausen“. In diesem Fall können alle Entitäten, die aus mindestens zwei Wörtern bestehen und nicht diese beiden Eigennamen enthalten, aus der Liste der möglichen Entitäten entfernt werden, wie z.B. „Michael Ballack“. Analog, wenn das vorhergehende Wort ein Eigenname ist (Zeile 13).
Anschließend wird ein Scoring durchgeführt, das entscheidet, welche die treffendste Entität ist (Zeile 14).

4.5.2 Scoring

Aufgabe des Scoring ist es, die korrekte Entität zu finden. Es werden dabei 3 verschiedene Faktoren verwendet, die das Scoring beeinflussen: Länge der Entität, Kontext und Anzahl der Vorkommen. Diese werden unterschiedlich gewichtet. Die Länge der Entität bildet den bedeutendsten Faktor, wohingegen Kontext und die Anzahl der Vorkommen versuchen das Ergebnis, unter jenen mit der besten Länge, in die richtige Richtung zu lenken.
Dies basiert darauf, dass wir grundsätzlich die simpelste Entität als am wahrscheinlichsten betrachten. Somit gilt: Je weniger Worte eine Entität hat, umso wahrscheinlicher sollte sie sein. Um die Punktzahl des Kontextes zu ermitteln, werden die genannten YAGO-Facts benutzt. Dazu werden die YAGO-Facts aller bereits gefundenen Entitäten in einen Trie gespeichert und anschließend für jeden Teilpfad der YAGO-Facts der aktuellen Entität überprüft, ob dieser im Trie enthalten ist. Die Anzahl der gemeinsamen Pfade mit den gefundenen Entitäten stellen schließlich die Kontextpunktzahl dar. Dabei lässt man die ersten Ebenen, ausgehend vom Wurzelknoten, weg, da diese zu abstrakt sind und das Ergebnis verfälschen. Denn unter Einschluss dieser, würde Kontext erkannt werden, der nicht vorhanden ist. Für die Anzahl der Vorkommen nutzen wir die in der Vorverarbeitung erstellten Liste.
Bevor die tatsächliche Punktzahl einer Entität berechnet wird, werden diese drei Faktoren auf eine maximale Punktzahl skaliert. Dies geschieht, da z.B. die Anzahl der Vorkommen größer 10 000 sein könnte, aber auch 0. Die Skalierung ermöglicht somit eine leichtere Gegenüberstellung der einzelnen Faktoren.
Zusätzlich können Entitäten Boni erhalten. So erhält eine Entität, die im gesuchten Wort enthalten oder mit dem gesuchten Wort sogar gleich ist, je nach Ähnlichkeit

einen großen Bonus. Weiterhin gilt im Falle von Disambiguierungen, dass diese einen umso größeren Bonus erhalten, je weiter vorne sie im Vektor stehen. Es werden nun die Punktzahlen aller möglichen Entitäten berechnet und anschließend wird jene mit den meisten Punkten gewählt.

5 Evaluation

Um eine Übersicht über die korrekt und falsch gefunden Entitäten zu bekommen, wurde eine Evaluation implementiert, die sowohl eine prozentuale Übersicht über das Ergebnis, sowie eine graphische Auswertung in HTML-Form bietet. Um die gefunden Entitäten eines Artikels zu evaluieren, ist es notwendig eine sogenannte Ground Truth zu übergeben. Bei einer Ground Truth handelt es sich um eine Datei, die das perfekte Ergebnis widerspiegelt und manuell erstellt werden muss. Die Ground Truth wird anschließend mit dem tatsächlichen Ergebnis verglichen, um einen Überblick über die Fehlerrate bekommen zu können.
In diesem Abschnitt wird zuerst auf den Aufbau der Ground Truth eingegangen. Folgend beschäftigen wir uns mit dem Auswertungsverfahren, um schließlich einen Überblick über die tatsächlichen Resultate zu bekommen und diese zu diskutieren.

5.1 Aufbau einer Ground Truth

Ziel der von uns erstellten Ground Truth war es effizient und mit möglichst wenig Information die Ergebnisse unserer Entitätenerkennung auswerten zu können. Wir entschieden uns dabei für folgendes Format:

<Wort> <PosTag> <korrekte Entität>

Wenn keine passende Entität existiert, so wird diese weggelassen.
Zur Veranschaulichung zwei Beispiele:

Merkel	EIGENNAME	Angela Merkel
ist	VERB	

Die PosTags sind in der Ground Truth enthalten, um ein zusätzliches, zeitintensives PosTagging vermeiden zu können.
Bei der Evaluation wird aus der Ground Truth ein Artikel generiert, auf welchem anschließend die Entitätenerkennung angewendet wird. Würde das PosTagging erst nach der Generierung des Artikels stattfinden, müssten während der Entitätenerkennung die PosTags wieder entfernt werden und Anfangs- und Endartikel könnten differieren, was zu einer höheren Fehleranfälligkeit führen würde.
Um das Format des Textes korrekt beibehalten zu können, werden Satzzeichen behandelt wie Wörter. Sie besitzen jedoch weder einen PosTag noch eine Entität.

5.2 Evaluation einer Ground Truth

Die Evaluation generiert aus der Ground Truth einen Artikel und speichert die gewünschten Entitäten. Anschließend wird auf dem Artikel eine Entitätenerkennung ausgeführt und die gefundenen Entitäten mit den gewünschten Entitäten verglichen. Dabei werden die Entitäten klassifiziert. Es existieren vier verschiedene Klassen:

Unmögliche Entitäten sind jene Entitäten, die nicht oder nur falsch gefunden werden können, da keine passende Entität in dem Trie vorhanden ist.

Falsch negative Entitäten sind jene Entitäten, für die, trotz gewünschter und möglicher Entität, keine Entität gefunden werden konnte.

Falsch positive Entitäten sind jene Entitäten, für die eine falsche Entität gefunden wurde und jene Entitäten, für die tatsächlich keine Entität gefunden werden sollte, da es sich um Pronomen, Präpositionen, etc. handelt.

Richtig positive Entitäten sind jene Entitäten, die den gewünschten Entitäten entsprechen. Nach der Klassifizierung werden dann die Trefferquote und Genauigkeit berechnet und ausgegeben (Meinzer, 2011).

Unter der Trefferquote versteht man den Anteil der richtigen Entitäten unter der Gesamtheit der falsch positiven und falsch negativen Entitäten. Die Genauigkeit steht hingegen für den Anteil für den überhaupt eine Entität gefunden wurde.
Zusätzlich wird der Anteil der unmöglichen Entitäten gegenüber den gesamten Entitäten ausgeben.

Um einen besseren Überblick über die Einzelheiten der Evaluation zu bekommen, wurde eine graphische Darstellung implementiert.

Abbildung 5.1: Graphische Evaluation eines Artikels

Bei dieser graphischen Auswertung wurden die Ampelfarben genutzt, um auf den Grad des Fehlers hinzuweisen. So bedeutet grün, dass die richtige Entität erkannt wurde, gelb, dass eine falsche Entität erkannt wurde und rot, dass gar keine Entität erkannt wurde, obwohl es eine passende Entität gab. Mit der Farbe blau, werden zwei verschiedene Arten von unmöglichen Entitäten gekennzeichnet. So finden wir unter dieser Farbe zum einen Entitäten, für die eine Entität gefunden wurde, obwohl es keine passende Entität gab. Diese verlinken auf den falsch gefundenen Artikel. Weiterhin werden mit blau jene unmöglichen Entitäten markiert, für die auch keine Entität gefunden wurde. Handelt es sich um Entitäten, die nicht gefunden hätten werden sollen, so werden diese orange markiert.

6 Resultate und Diskussion

Ziel dieses Abschnittes ist es die Resultate der Evaluation vorzustellen und zu interpretieren, sodass jene Probleme verstanden werden, die zu einer Verschlechterung des Ergebnis führen. Vor der Diskussion wird zuerst die Evaluation selbst interpretiert, um auf ihre Grenzen aufmerksam zu machen. Anschließend werden die Ergebnisse evaluiert. Diese lassen sich einteilen in unmögliche Entitäten, Genauigkeit und Trefferquote, auf die in gleicher Reihenfolge eingegangen wird.
Für die Evaluation wurde die Entitätenerkennung auf drei verschiedenen Ground Truths getestet, die manuell erstellt wurden. Die Resultate sind in Abbildung 6.1 zu finden.

Ground Truth	Trefferquote in %	Genauigkeit in %	unmögliche Entitäten in %
1. Ground Truth[1]	90, 1	97, 1	14, 2
2. Ground Truth[2]	86, 3	97, 6	16, 6
3. Ground Truth[3]	75, 8	97, 0	27, 3

Abbildung 6.1: Resultate der Evaluation auf drei verschiedenen Ground Truths

Da nur drei Ground Truths existieren, ist das Ergebnis nur bedingt repräsentativ. Es erlaubt einen Überblick über die Effizienz der Entitätenerkennung zu bekommen und hilft Probleme zu identifizieren. Wenn im weiteren von der mittleren Trefferquote, der mittleren Genauigkeit und den mittleren Anteil unmöglicher Entitäten gesprochen wird, so bezieht sich dies auf den Mittelwert dieser drei Resultate aus den Ground Truths. Die Werte können natürlich im Vergleich zu dem Mittel mit mehr Ground Truths differieren.

6.1 Interpretation der Evaluation

Bevor wir zu den eigentlich Resultaten kommen, ist es wichtig die Probleme der Evaluation zu verstehen, die das Ergebnis verfälschen können.
Ein Problem der Evaluation besteht darin, dass es nur eine perfekte Entität geben

[1] http://stromboli.informatik.uni-freiburg.de/baumgari/GroundTruth/sz01.tagged.html
[2] http://stromboli.informatik.uni-freiburg.de/baumgari/GroundTruth/sz02.tagged.html
[3] http://stromboli.informatik.uni-freiburg.de/baumgari/GroundTruth/sz03.tagged.html

kann. In vielen Fällen lässt sich nicht eindeutig sagen, welche Entität die Korrekte ist. Man betrachte das Beispiel des elektrischen „Strom“s im alltäglichen Wortgebrauch. „Strom“ ist eine Disambiguität und muss daher aufgelöst werden. Die deutschsprachige Wikipedia bietet hier mehrere sinnvolle Möglichkeiten: „Elektrischer Strom“, „Elektrizität“, „Fluss (Physik)“. Alle diese Bedeutungen machen in diesem Zusammenhang für Strom Sinn und sollten als korrekt eingestuft werden. Diese Möglichkeit hat die Evaluation allerdings bis dato noch nicht. Aus diesem Grund kann die Trefferquote sinken, obwohl kein echter Fehler existiert.
Weiterhin werden auch mehrnamige Entitäten, wie z.B. „Martin Günthner“ als zwei unmögliche Entitäten betrachtet. Aus diesen Gründen kann, in Abhängigkeit des Artikels, der prozentuale Anteil der unmöglichen Entitäten zu hoch sein.
Zusätzlich sollte bedacht werden, dass die Ground Truth manuell erstellt wurde und daher übersehene Fehlern innerhalb der Ground Truth vorkommen können, die das Ergebnis verfälschen. Schließlich sollte man sich dessen bewusst sein, dass die unmöglichen Entitäten, nicht mit in die Trefferquote und die Genauigkeit einfließen.

6.2 Unmögliche Entitäten

Die unmöglichen Entitäten nehmen im Mittel etwa 19 % der im gesamten gesuchten Entitäten ein und machen damit einen sehr großen Teil des Textes aus.
Es gibt drei verschiedene Gründe, die zu unmöglichen Entitäten führen.
Zwei dieser Gründe, die zudem den Großteil der unmöglichen Entitäten ausmachen, lassen sich bei Verwendung mit der Wikipedia nicht vermeiden.

Mapping auf englischsprachige Wikipedia In vielen Fällen fehlen Verlinkungen deutscher Artikel auf die englischsprachige Wikipedia. Ein Beispiel hierfür ist der Begriff „Bündnis“. Da dieser Begriff keine Verlinkung auf die englischsprachige Wikipedia besitzt, wird er im Trie nicht gespeichert und kann damit nicht gefunden werden.

Fehlende Entitäten in der deutschsprachigen Wikipedia Es existieren nicht für alle Begriffe passende Artikel innerhalb der deutschsprachigen Wikipedia, wie z.B. der Begriff „Anfang“. Dieser Begriff ist in der Wikipedia nicht enthalten, denn „Wikipedia ist kein Wörterbuch“[4]. Aus diesem Grund ist er auch nicht im Trie enthalten und kann nicht gefunden werden.

Fehlerhaftes Parsing Ein weiterer Grund sind Fehler beim Auslesen der Weiterleitungs-, Disambiguierungs- und Artikelseiten aus der Wikipedia. So wird z.B. „Deutschland“ beim Parsen der Disambiguierungsseite „Bund“ in der XML-Datei

[4] http://de.wikipedia.org/wiki/Wikipedia:Was_Wikipedia_nicht_ist

nicht mit aufgenommen. Dies geschieht dadurch, dass die XML nicht vollkommen standardisiert ist. In der XML-Datei gibt es Tags mit verschiedenen Bedeutungen, wie z.B. Auflistung, Link innerhalb der Wikipedia, Weiterleitungen. Allerdings gibt es oft mehrere Tags mit den gleichen Bedeutungen. Als Beispiel gibt es z.B. für Weiterleitungen mehrere Tags die auf Weiterleitungen hinweisen (#Redirects, #Weiterleitungen, etc.). Auch die Klammerung von Links ist nicht immer identisch. Bisher sind nicht alle Sonderfälle eingeschlossen, was zu fehlenden Entitäten führt.

Konsequenzen unmöglicher Entitäten Die Problematik unmöglicher Entitäten besteht darin, dass sinnlose Entitäten gefunden werden, die das Suchergebnis später verfälschen können. Ein Beispiel dessen ist, dass dadurch schnell Entitäten wie Filme, Buchtitel und Serien gefunden werden. Diese bestehen meist aus mehreren Wörtern und es ist wird dadurch wahrscheinlich, dass sie eine mögliche Entität darstellen. Aus Mangel passenderer Entitäten werden diese dann fälschlicherweise ausgewählt. Ein Beispiel hierfür wäre der Film „Auf Anfang“, wenn die Entität „Anfang“ gefunden werden soll.

6.3 Genauigkeit

Die Genauigkeit liegt im Mittel bei etwa 97%.
Dies lässt sich damit begründen, dass die Entitätenerkennung eher zu viel findet als zu wenig. Denn die Entitätenerkennung sortiert nur jene Begriffe aus, die keinesfalls Entitäten darstellen können, wie unwichtige Wörter und Wörter der falschen Wortart, wodurch alle möglichen Entitäten betrachtet werden. Die fehlende Genauigkeit kommt aus drei verschiedenen Gründen zustande.

Kompositazerlegung Der zweite Grund liegt in der Zerlegung der Komposita. Die Ground Truth enthält Wörter, die sich zwar teilen lassen, aber von welchen nicht beide Teilbegriffe von der Entitätenerkennung erkannt werden können, da nicht beide Entitäten existent sind. Ein Beispiel wäre hierfür in der 1. Ground Truth das Wort „Wirtschaftsmisere“. Die Entität „Wirtschaft“ existiert, „Misere“ jedoch nicht.

Stemming Der letzte Grund für fehlende Entitäten ist das Stemming. Viele Wörter ändern im Plural einen Vokal in ein Umlaut, wie „Baum - Bäume“, „Hand - Hände“. Dies wird vom Stemming-Algorithmus bisher nicht beachtet, wodurch die Entitäten nicht gefunden werden können. Der nächste Grund ist, dass nicht alle möglichen Endungen bedacht werden. So werden auch Diminuitive, wie „Kind - Kindchen“, „Hand - Händchen“, bisher nicht beachtet.

Falsche Wortart Der PosTagger weist 96% der Begriffe die richtige Wortart hinzu. Wird von ihm ein Substantiv nicht als Substantiv erkannt, so wird diese Entität nicht erkannt werden. Dies kommt jedoch sehr selten vor.

6.4 Trefferquote

Die mittlere Trefferquote liegt bei etwa 84% und schwankt sehr in Abhängigkeit des Textes. So ist die Trefferquote für die 3. Ground Truth mit 75% wesentlich schlechter als für die 1. Ground Truth, welche eine Trefferquote von $90,1$ Prozent erreicht.
Im Folgenden sollen an der 3. Ground Truth nun zuerst die zwei Hauptprobleme erklärt werden, die zu einem sehr schlechten Ergebnis führen können, um anschließend auf weitere Probleme einzugehen. Der Artikel beschäftigt sich mit Atomkraft und ist sehr eng gefasst, sodass viele Aufzählungen und Wiederholungen vorkommen.

6.4.1 Unzureichende Kontexterkennung

Bei Betrachtung der 3. Ground Truth stellt man fest, dass das erheblichste Problem in falsch erkannten Entitäten liegt. Oft werden die Begriffe so erkannt, dass sie in einem anderen Kontext korrekt wären, wie „Krümmel“ als „Krümmel (Westerwald)“ anstatt als „Kernkraftwerk Krümmel“. Der Grund hierfür liegt in der Kontexterkennung.
Diese führt innerhalb des Scoring zu erheblichen Problemen. Diese Probleme basieren im wesentlichen auf zwei Gründen: die aus YAGO abgeleiteten YAGO-Facts sind zu ungenau und unvollständig, um den Kontext korrekt zu setzen. Die Unvollständigkeit begründet sich darauf, dass die YAGO-Facts bereits älter sind und neue Artikel hinzukamen, die zum Zeitpunkt der Erstellung der YAGO-Facts noch nicht enthalten waren. Viele Relationen werden also nicht erkannt, da hierfür die Einträge fehlen. So fehlt z.B. der Pfad für „Kernkraftwerk Lingen“, wodurch dieses mit „Kernkraftwerk Krümmel“ keine Gemeinsamkeiten hat.
Zudem sind die Pfade selbst oft ungenau. Zum Beispiel besitzt nicht nur das „Kernkraftwerk Krümmel“ den Pfad „entity:physicalentity:object:whole:artifact:facility :station“, sondern ebenso die „Berliner U-Bahnstation Schönleinstraße“. Zwei Entitäten, die nur sehr fern einen gemeinsamen Kontext besitzen.
Ein weiteres Problem stellt die Überrepräsentation von Orten dar. In der Wikipedia sind die meisten Orte Deutschlands von Dörfern bis Großstädten zu finden, die zusätzlich meist mit mehreren Namen gespeichert sind. Betrachtet man ausschließlich die Disambiguierungsseite Grund[5] so stellt man fest, dass es sehr viele Orte als Bedeutung gibt. Im normalen Sprachgebrauch sind diese jedoch nur selten gemeint. So wird für „Grund“ im Sinne von Kausalität oft „Wullersdorf“ erkannt. Da

[5] http://de.wikipedia.org/wiki/Grund

grundsätzlich alle Bedeutungen gleich wahrscheinlich sind, stimmen hier die Relationen im Vergleich zum deutschen Sprachgebrauch nicht mehr. Denn hier sind meist eher abstraktere Begriffe wie „Kausalität“ gemeint und selten Orte.

6.4.2 Kettenreaktionen

Die Entitätenerkennung uneindeutiger Entitäten, geht davon aus, dass, wenn ein Teil einer bereits gefundenen Entität vorkommt, diese mit aller Wahrscheinlichkeit auch gemeint ist. Ein meist positives Beispiel hierfür wäre „Angela Merkel“. Finden wir im folgenden Text „Merkel“, so ist diese wahrscheinlich gemeint. Selbiges gilt auch für vollständige Entitäten, wie „Haus“. Wurde diese Entität einmal als „Haus“ erkannt, so sollte diese auch wieder als „Haus“ erkannt werden. Dies hat den Grund, dass sich die Bedeutungen innerhalb eines Textes nur selten ändern. Durch dieses Verfahren können nun, je nach Artikel, immense Folgefehler auftreten, denn nicht immer ist die gefundene Entität die gewünschte Entität. Da in der 3. Ground Truth sich sehr viele Wörter wiederholen, ist die Problematik hier sehr ausgeprägt und es kommen viele solcher Kettenreaktionen zustande, die die Trefferquote rapide verschlechtern. Im Folgenden soll nun erklärt werden, wie es zu diesen Fehlern genau kommt. Es lassen sich dabei verschiedene Arten von Kettenreaktionen benennen.

Falsch erkannte Entitäten Der häufigste Fehler ergibt sich durch falsch erkannte, sich wiederholende Entitäten. Ein Beispiel aus der 3. Ground Truth wäre „Netz“. Dieses Wort wurde zu Beginn als „Netz (Sternbild)“ erkannt, anstatt als „Stromnetz“. Da es viermal innerhalb des Textes vorkommt, verschlechtert sich die Trefferquote. Dieses Problem tritt recht häufig auf und zeigt sich ebenso in der zweiten Ground Truth („Bund als Staat anstatt Gliedstaat“), wie auch („Jahren als Anne Jahren anstatt Jahr“).

Kettenreaktion durch unzureichende Kontexterkennung Wie im Unterabschnitt unzureichende Kontexterkennung erklärt, lassen sich für einen sehr großen Teil der Entitäten Orte finden. Nun besteht die Problematik im Folgenden: wurde ein Ort zuviel erkannt, wird die Wahrscheinlichkeit wieder einen Ort zu finden noch größer, da die entstehende Kontextpunktzahl für einen Ort sehr schnell die der anderen Entitäten übertrumpft. Dadurch wird bei der nächsten Möglichkeit wieder ein Ort erkannt. Es entsteht also eine Kettenreaktion. Bei Orten ist dieser Folgefehler aufgrund der Überepräsentation sehr stark ausgeprägt.

Gleiche Wörter mit verschiedenen Bedeutungen Bei der Erkennung der eindeutigen Entitäten wurde in der 3. Ground Truth „Fukushima 1“, auch „Kernkraftwerk Fukushima-Daiichi“ genannt, gefunden. Nun ist es in dem Artikel der Fall, dass das Wort „Kernkraftwerk“ mehrmals vorkommt. Aufgrund des genannten Verfahrens, wird nun in jedem Fall das „Kernkraftwerk Fukushima-Daiichi“ erkannt. Das

Problem hier liegt in der Annahme, dass es sich auf jeden Fall wieder um dieselbe Entität handeln wird.

Es wird ersichtlich, dass die Ursache der meisten Probleme in der unzureichenden Kontexterkennung liegt. Da dieses Problem allerdings nur schwer zu lösen ist, sollten Mechanismen gefunden werden, die das daraus folgende Problem der Kettenreaktionen möglichst minimieren.

6.4.3 Weitere Probleme

Die bisher genannten Probleme sind jene mit dem größten Einfluss auf Trefferquote und Genauigkeit. Dennoch gibt es noch weitere Probleme die es zu beachten gilt. Diese sollen in diesem Abschnitt erklärt werden.

Disambiguierungsseiten Oftmals sind die Disambiguierungen nicht entlang ihrer häufigsten Bedeutung geordnet, sondern nach Themenbereichen, wodurch sich Fehler bei den Entitäten einschleichen. Man nehme als Beispiel das Wort „Netz“[6] aus der 3. Ground Truth. In den meisten Fällen ist hier ein Begriff aus dem Bereich Technik gemeint, der allerdings aufgrund seines niedrigen Bonus nicht gewinnt.
Weiterhin sind nicht alle Entitäten im Trie, die einer Disambiguierungsseite entsprechen, auch als Disambiguierungsseite gespeichert. Ein Beispiel hierfür wäre die Entität „Bilder“ aus der 2. Ground Truth, ein Artikel, der auf die Disambiguierungsseite „Bild“ weiterleitet. Der Grund hierfür liegt darin, dass wir zuerst die Weiterleitungs- und anschließend die Disambiguierungsseiten einlesen. Auf den Weiterleitungsseiten ist nicht gespeichert, ob das Ziel eine „Disambiguierungsseite“ ist. Aus diesem Grund weiß die Weiterleitung nicht, dass der Link zu einer Disambiguierungsseite führt und speichert die Entität als normale Entität ohne Disambiguierungen. Die Liste der Disambiguierungen weiß auch nichts von dem Eintrag der Weiterleitung und kann diese damit nicht als Disambiguierung markieren. Dadurch wird die Entität „Bilder“ nicht als Disambiguierungsseite gespeichert und kann nicht aufgelöst werden.

Stemming Es gibt keine allgemeine Regel, wie lang Endungen in Abhängigkeit des Wortes sind. Aus diesem Grund wird in manchen Fällen zuviel entfernt, sodass aus „Minute“ die Entität „Minu“ entstehen kann. Ein weiteres Problem des Stemming ist, dass bisher Umlaute nicht beachtet werden. So ist die „Plätze“ die Mehrzahl von „Platz“. Es kann jedoch lediglich „Plätz“ erkannt werden.

Deklination Das implementierte Stemming bietet die Möglichkeit Wörter auf ihre Stammform zurückzuführen, insofern sie „zuviele“ Buchstaben enthalten. In vielen

[6] http://de.wikipedia.org/wiki/Netz

Fällen wird ein Wort jedoch an den Kasus angepasst und Buchstaben getauscht. Man nehme als Beispiel „Internationalem Währungsfond“. In diesem Beispiel müsste „Internationalem“ auf „Internationaler“ umgewandelt werden, um die Entität zu erkennen.

Kompositazerlegung In manchen Fällen werden Wörter in unsinnige Entitäten zerlegt. So wird zum Beispiel in der 1. Ground Truth „Voraussetzung“ in den Ort „Vorau“ und „setzung“ geteilt.

Datenstruktur Trie Der verwendete Trie ist effizient, es folgen aus ihm jedoch auch Nachteile. So sucht der Trie unabhängig von Groß- und Kleinschreibung, was zur Folge hat, dass manche Entitäten überschrieben werden. So existieren in der Wikipedia sowohl „Kopf“[7] als auch „KOPF“[8].„Kopf“ wird innerhalb des Tries einfach überschrieben und kann in Folge dessen nicht mehr gefunden werden. Ein weiterer Nachteil dieser Datenstruktur ist der Mangel an Flexibilität. So kann unsere Entitätenerkennung nur Teilentitäten erkennen, die am Ende oder am Anfang einer Entität vorkommen. Zum Beispiel ist die Entitätenerkennung nicht in der Lage „Karl-Theodor zu Guttenberg“[9] zu erkennen, wenn lediglich „Theodor zu Guttenberg“ gegeben ist.

[7] http://de.wikipedia.org/wiki/Kopf
[8] http://de.wikipedia.org/wiki/KOPF
[9] http://de.wikipedia.org/wiki/Karl-Theodor zu Guttenberg

7 Ausblick

Ziel dieser Arbeit war es eine Entitätenerkennung vorzustellen, die dazu in der Lage ist wichtigen Begriffen in Zeitungsartikeln die korrekte Bedeutung zuzuordnen. Um dies zu erreichen wurde Vorwissen generiert, welches die möglichen Entitäten, YAGO-Facts und die Artikel mitsamt Eigenschaften und Wortarten der einzelnen Wörter enthält. Um die Entitäten erkennen zu können, wurde der Prozess in zwei Teile aufgeteilt: der Erkennung der eindeutigen Entitäten und der Erkennung der uneindeutigen Entitäten. Mit Hilfe von verschiedenen Verfahren wie Scoring, Pos-Tagging, Kontexterkennung, Kompositazerlegung und Stemming wurden versucht die korrekten und alle möglichen Entitäten zu ermitteln.
Die implementierte Entitätenerkennung wurde evaluiert und das Ziel wurde wurde mit einer zufriedenstellenden Genauigkeit von etwa 97 %, einer Trefferquote von etwa 84 % erreicht und einem Anteil unmöglicher Entitäten von etwa 19 % erreicht. In diesem Kapitel soll nun ein Ausblick auf mögliche Lösungen zu den in der Diskussion genannten Problemen gegeben werden. Die Lösungsansätze werden in absteigender Reihenfolge entlang ihrer Priorität sortiert:

1. YAGO-Facts:
   Mehr dazu unter Unzureichende Kontexterkennung, Kettenreaktionen.
   Das Problem der Unvollständigkeit der YAGO-Facts ließen sich durch ein erneutes Ableiten von YAGO-Facts aus einer aktuelleren YAGO-Version lösen. Dennoch sollte hier bedacht werden, dass die Ungenauigkeit bestehen bleiben wird, sodass sich die Kontexterkennung nur minimal verbessern wird. Dieses Problem ließe sich nur durch das Nutzen einer alternativen Ontologie lösen, die für diese Aufgabenstellung besser geeignet ist. Die Ontologie sollte dabei folgende Kriterien erfüllen: sie sollte auf den Entitäten der Wikipedia aufbauen und der Kontext sollte genauer spezifiziert werden können. Durch Verbesserung der Kontexterkennung würde sich auch die Anzahl der Kettenreaktionen aufgrund eines falsch aufgebauten Kontextes (z.B. durch falsch erkannte Orte) minimieren. Auch das Problem der nach Themenbereichen sortierten Disambiguierungsseiten, ließe sich damit lösen. Die Verbesserung der Kontexterkennung ist unter allen Problem das wichtigste, aber auch zeitintensivste und schwierigste Problem.

2. Minimierung von Kettenreaktionen:
   Mehr dazu unter Kettenreaktionen.
   Das Problem der Kettenreaktionen aufgrund falsch erkannter Entitäten lässt sich minimieren, indem erst während des Scorings auf bereits gefundene En-

titäten geprüft wird und nicht davor. Entitäten die bereits gefunden wurden, sollten dabei einen Bonus kriegen. Der Vorteil dieses Verfahrens läge darin, dass auch die anderen möglichen Entitäten betrachtet werden und gewinnen können. Zudem lässt sich diese Änderung schnell bewerkstelligen. So würde wahrscheinlich „Kernkraftwerk“ gegenüber „Kernkraftwerk Fukushima-Daiichi“ gewinnen, da es weniger spezifisch ist.

3. Minimierung unmöglicher Entitäten:
   Mehr dazu unter Unmögliche Entitäten.
   Durch die unmöglichen Entitäten kommen viele fehlerhaft erkannte Entitäten zustande, welche minimiert werden sollten um das Suchergebnis weniger zu verfälschen. Eine Lösungsansatz für dieses Problems wäre die Einführung einer Grenze innerhalb des Scoring. Fällt die Punktzahl der besten Entität unter diese Grenze, so sollte keine Entität ausgewählt werden. Vorteil dieses Verfahrens wäre die Minimierung der unmöglichen Entitäten und die leichte Implementation. Zu bedenken ist hierbei allerdings, dass auch die Genauigkeit etwas sinken wird. Dies hat den Grund, dass damit auch manche korrekte Entitäten nicht mehr erkannt werden können, da ihre Punktzahl zu niedrig ist. Dies sollte im Vergleich allerdings eher gering sein.

4. Minimierung von fehlerhaft erkannten Titeln:
   Mehr dazu unter Unmögliche Entitäten.
   Die fälschliche Erkennung von Buchtiteln, Filmtiteln und Serien in den (meist) unmöglichen Entitäten ließe sich leicht und schnell lösen, indem Entitäten, die vollständig vorkommen müssen, identifiziert werden. Dies ließe sich bewerkstelligen, indem die YAGO-Facts geprüft werden. Handelt es sich laut YAGO-Fact bei der aktuellen Entität um den Titel eines Buches, eines Films oder einer Serie, so sollte diese Entität nur gewählt werden, wenn sie vollständig ist. Dies führt allerdings zu einem spezifischeren, unflexibleren Code.

5. Entitätengewinnung durch Verbesserung des Trie:
   Mehr dazu unter Weitere Probleme.
   Um das Problem der fehlenden Entitäten aufgrund des von der Groß- und Kleinschreibung unabhängigen Einfügens und Auslesens zu lösen, könnte man den Trie dementsprechend abändern, dass Entitäten in Abhängigkeit von Groß- und Kleinschreibung einfügt, jedoch unabhängig von der Schreibweise gesucht werden. In diesem Fall müsste anstatt einer gefundenen Entität eine Menge gefundener Entitäten zurückgegeben werden. Dies würde natürlich die Effizienz des Tries beeinflussen. Der negative Einfluss sollte sich jedoch in Grenzen halten, da die Problematik nur selten auftritt. Zudem sollte sich der zeitliche Aufwand der Implementation im Rahmen halten.

6. Verbesserung der Evaluation:
   Mehr dazu unter Interpretation der Evaluation.
   Die Evaluation sollte so erweitert werden, dass es mehrere perfekte Entitäten geben kann. Dies lässt sich leicht realisieren, indem nach der 3. Spalte, die

aktuell die gewünschte Entität darstellt, auch weitere Spalten mit weiteren gewünschten Entitäten hinzugefügt werden. Somit hätte die Ground Truth anschließend folgende Struktur:
<Wort> <PosTag> <korrekte Entität> <korrekte Entität> ...

7. Erweiterung des Stemming:
   Mehr dazu unter Weitere Probleme.
   Genauigkeit und Trefferquote ließen sich durch eine Erweiterung das Stemming verbessern. So sollte während des Stemmings versucht werden einzeln die Umlaute in den dazugehörigen Vokal umzuwandeln „Plätze - Platz“. Weiterhin sollten auch verschiedene Deklinationen in Betracht gezogen werden („Internationalem - Internationaler“). Da diese Regeln allgemein definiert werden können, lässt sich der Stemming-Algorithmus dahingehend problemlos erweitern. Zusätzlich müsste hier allerdings auch das Stemming innerhalb einer Phrase implementiert werden, da es bisher nur auf Wörtern und nicht auf Phrasen läuft.

8. Korrekturen beim Parsen der Wikipedia:
   Mehr dazu unter Unmögliche Entitäten.
   Das Parsen der Wikipedia-XML sollte erweitert werden, sodass jedmögliche Sonderfälle eingeschlossen sind.

9. Verbesserung der Kompositazerlegung:
   Mehr dazu unter Weitere Probleme.
   Ein Ansatz um weniger falsche Komposita zu erkennen, ist die Einführung einer Liste mit typischen Suffixen und Präfixen, wie „ab“, „vor“, usw.. Der Nachteil darin besteht natürlich in der Notwendigkeit eines noch größeren Vorwissens.

10. Verbesserung in der Erkennung von Teilentitäten:
    Mehr dazu unter Weitere Probleme.
    Da in seltenen Fällen die zwei Permutationen der Entität nicht ausreichen, wie im Beispiel der Suche nach „Theodor zu Guttenberg“ für den ausschließlich „Karl-Theodor zu Guttenberg“ und „Guttenberg zu Karl-Theodor“ gespeichert ist, könnte man alle möglichen, sinnvollen Permutation speichern: „Karl-Theodor zu Guttenberg“, „Theodor zu Guttenberg Karl“, „zu Guttenberg Karl-Theodor“, „Guttenberg Karl-Theodor zu“. Der Nachteil dieses Verfahrens wäre jedoch der immense Platzbedarf. Ein Alternativansatz wäre die Einführung eines „Egal“-Zeichens, sodass im Baum alle möglichen Wege abgesucht werden und alle Entitäten zurückgegeben werden, die die gesuchte Zeichenkette enthalten. Dadurch würde sich die Komplexität der Suche sehr erhöhen, jedoch würde sich das Problem des Platzbedarfs nicht verschlechtern.

Ein weitere Schritte sollte sein, die Entitätenerkennung nicht nur auf Nomen laufen zu lassen, sondern ebenso auf Adjektiven. Inbesondere adjektivierte Nomen und Eigennamen, wie „furcht-bar“ und „Newton-sche“, sind in vielen Fällen Quellen wichtiger Information. Die bisherigen Verfahren eignen sich noch nicht sehr gut für die

Entitätenerkennung in Adjektiven, doch sollte dies mit einer Erweiterung des Stemmingverfahrens möglich sein. So müsste z.B. für das adjektivierte Nomen „häuslich“ die Endung „lich“ entfernt und der Umlaut „ä“ in „a“ umgewandelt werden.
Weiterhin könnten die nächsten Schritte sein, diese Entitätenerkennung auch auf eine größere Menge von Artikeln zu testen und ebenso auf Fachtexten zu erproben. Schließlich könnte man versuchen die Entitätenerkennung auch für andere, ähnlich aufgebaute, Sprachen zu erweitern. Dazu müssten die Artikel in der jeweiligen Sprache gepostaggt und die sprachspezifische XML-Datei der Wikipedia neu eingelesen werden. Zusätzlich sollte das Stemming auf die jeweilige Sprache angepasst werden. Da in vielen Sprachen keine Komposita durch Zusammenschreiben zweier Substantive, wie im Deutschen, existieren, kann die Kompositazerlegung im besten Fall sogar weggelassen werden. Ansonsten sollte auch diese angepasst werden.

Danksagung

An dieser Stelle möchte ich meinen Dank insbesondere an Prof. Dr. Hannah Bast und meinem Betreuer Björn Buchhold richten, die mich über den gesamten Zeitraum der Bachelorarbeit hinweg wirklich sehr unterstützt und mir zu vielen Versionen meiner Bachelorarbeit wunderbares Feedback gegeben haben.

Weiterhin möchte ich mich bei Katrin Drevin bedanken, die mir auch in den Zeiten, in denen mir alles über die Ohren wuchs, zur Seite stand und mich unterstützt hat. Dafür werde ich mich auf jeden Fall revanchieren.

Schlussendlich will ich mich natürlich auch bei all meinen Korrekturlesern bedanken, die die Mühe auf sich genommen haben, die ganze Thesis durchzuarbeiten: Rebecca Albrecht, Sindy Baumgarten und Peer Roggendorf.

Vielen Dank euch allen.

Literaturverzeichnis

Buchhold, Björn: SUSI: Wikipedia Search Using Semantic Index Annotations. Masterarbeit, Albert-Ludwigs-Universität Freiburg, Lehrstuhl für Algorithmen und Datenstrukturen, 2010

Meinzer, Niklas: Semantische Suche in Zeitungsartikeln. Bachelorarbeit, Albert-Ludwigs-Universität Freiburg, Lehrstuhl für Algorithmen und Datenstrukturen, 2011

Schmid, Helmut: TreeTagger - a language independent part-of-speech tagger. 1994 ⟨URL: `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`⟩ – Zugriff am 21.09.2011

Schmid, Helmut: Improvements In Part-of-Speech Tagging With an Application To German. Dublin, 1995 ⟨URL: `http://www.ims.uni-stuttgart.de/ftp/pub/corpora/tree-tagger2.pdf`⟩ – Zugriff am 21.09.2011