

Disambiguierung - Lösen von Mehrdeutigkeiten

Marco Giccone, Patryk Gonsior, Erkan Sapmaz, Eva Widera

JW Goethe-University Frankfurt am Main, Dept. of Computer Science

Robert-Mayer-Str. 11-15, 60486 Frankfurt am Main, Germany

Email: m.giccone@web.de, trm@papego.com, esapmaz@web.de, widera_eva@hotmail.com

Abstract

Zunächst werden wir im ersten Kapitel eine Einführung in die Materie liefern und mögliche Formen der Ambiguität vorstellen. Kapitel 2 befasst sich dann mit der Geschichte, verschiedenen Verfahren und möglichen Anwendungsgebieten der Disambiguierung. Ausserdem wird ein Verfahren der Disambiguierung detaillierter vorgestellt, um zu zeigen, wie ein möglicher Ansatz in der heutigen Forschung aussehen kann. Der Hauptteil unserer Dokumentation befasst sich dann mit unseren Ansätzen und den aufgetretenen Problemen, Erfahrungen und Ergebnissen bei der Ausarbeitung und Implementierung unseres Programmes, das sich zum Einen auf benutzergesteuerte Eingaben stützt (supervised) und zum Anderen vollkommen automatisch arbeitet (unsupervised).

1 Einleitung

Der Ursprung des Wortes Ambiguität liegt im Latein (ambiguitas : Zweideutigkeit, Doppelsinn) und bezeichnet ein Zeichen, das mehrere Bedeutungen besitzt. Ambiguität tritt somit nicht zwingend in der Linguistik auf und kann zum Beispiel auch bei einem Bild auftreten, dass auf zwei verschiedene Weisen gedeutet werden kann, wie es oft in der optischen Täuschung auftritt. Jedoch werden wir nur das Auftreten von Mehrdeutigkeiten bei sprachlichen Zeichen in der Linguistik betrachten und versuchen diese aufzulösen, also zu disambiguieren. In der Linguistik entsteht demnach Mehrdeutigkeit, wenn ein Ausdruck (egal ob ein einzelnes Wort, eine Wortkombination oder ein ganzer Satz) in mehrfacher Weise interpretiert werden kann. Die auftretende Ambiguität kann ein Problem sein, dass in vielen Bereichen korrigiert werden sollte. Wenn man als Beispiel einen Gesetzestext betrachtet, wird hier akribisch darauf geachtet, Mehrdeutigkeiten zu vermeiden. Jedoch werden wir im nächsten Punkt des Kapitels sehen, dass Ambiguitäten auch absichtlich eingesetzt werden können.

Es gibt zahlreich verschiedene Formen von Mehrdeutigkeiten, welche sich in verschiedene Kategorien einteilen lassen. Zunächst wäre die syntaktische Ambiguität zu erwähnen, bei der eine Wortfolge (bis zu einem ganzen Satz) verschiedene Bedeutungen aufweisen kann. Damit eng verbunden steht die semantische Mehrdeutigkeit (hierbei hat ein Ausdruck

unterschiedliche Funktionen in der Gesamtbedeutung des Satzes) und die Pseudoambiguität, bei der sich die Bedeutung eines Ausdrucks im Verlauf des Satzes ändert.

Die zweite große Gruppe der Ambiguitäten ist die lexikalische Mehrdeutigkeit. Bei dieser Form ist nur ein Wort betroffen. Diese Ambiguität tritt häufig bei Namen, in der Fach- und Umgangssprache oder in der Werbung auf, wobei letztere wohl immer beabsichtigt auftritt, damit sich ein Produkt besser einprägt. Eine weitere Unterform der lexikalischen Ambiguität ist die Vagheit, die auftaucht, wenn die lexikalische Bedeutung nicht eindeutig durch den Inhalt definierbar ist. Als Beispiel lässt sich das Wort "Onkel" betrachten, bei dem entweder der Bruder der Mutter oder des Vaters gemeint sein kann. Eine Art von Mehrdeutigkeit, die häufig im alltäglichen Lesen von Texten auftritt ist die pragmatische Ambiguität, die bei der Verwendung von Pronomen oder anderen Abkürzungen auftreten kann. Als Beispiel betrachten wir den Satz: "Sie ließ das Glas auf das Tablett fallen und zerbrach es" ; hier ist nicht eindeutig, ob das Glas oder das Tablett zerbrochen ist.

Es lassen sich mit Sicherheit noch viele weitere Formen aufzählen, jedoch soll dies als kurzer Überblick genügen, da wir uns bei unseren Ansätzen auf die Lexikalische Mehrdeutigkeit beschränken wollen.

2 Related work

2.1 State of the art

Disambiguierung wird notwendig bei Mensch-Maschine-Schnittstellen, d.h. sobald für den Computer Natural Language Processing (NLP) Verfahren zur Anwendung kommen. Dabei wurden die Anwendungsgebiete mit der Zeit immer zahlreicher. Begonnen mit der automatischen Übersetzung und der Textanalyse kamen später Sprachsynthese (aus Texten Audiosignale erzeugen), Orthographie (Korrektur von Rechtschreib- und Grammatikfehlern) sowie Information Retrieval (Liefern von richtigen Antworten auf Anfrage) dazu. Das Auflösen von Ambiguitäten wird schon seit den ersten Computern Mitte des letzten Jahrhunderts erforscht. Zuerst wurde eher theoretisch Polysemie analysiert. So hat Margaret Masterman 1957 anhand von Nummernübereinstimmung des Wortstammes in Roget's Thesaurus das lateinische

Original von Vergils *Georgica* ins Englische übersetzt. Doch insgesamt kam die Disambiguierung nicht wirklich voran. Ursache war die Disambiguierung selbst. Beispielhaft dafür ist der Satz "the box is in the pen" (Yohoshua Bar-Hillel), der ohne Wissen über den Kontext nicht automatisch eindeutig aufgelöst werden kann (pen = Stift oder pen = Laufstall). Zwischenzeitlich wurde versucht durch Übersetzung in andere "Sprachen" (hauptsächlich Mathematik) Bedeutungen aufzulösen, jedoch ohne Erfolg. Erst durch Verfahren aus der Künstlichen Intelligenz (KI) wurde die Forschung in der Word Sense Disambiguation (WSD) wieder intensiver. Eine KI besteht aus einem Netzwerk in dem zwischen Knoten Signale ausgetauscht werden. Die Verfahren in diesem Bereich teilen sich in zwei Gruppen auf: die erste Gruppe stellen symbolische Verfahren dar: diese Verfahren wurden zuerst entwickelt; jedem Knoten wurde ein abstraktes Konzept für die Wörter zugeordnet, wie z.B. "eßbar" / "ungenießbar", "sicher" / "unsicher". Das Weiterleiten von Signalen findet hier durch "marker-passing" statt, d.h. anhand von Regelsätzen werden nun Konzepte aktiviert, wenn passende andere Konzepte/Konten vorher aktiviert wurden. Die andere Gruppe stellen "subsymbolische" bzw. "konnektionistische" Verfahren dar. Zentraler Unterschied zu den erstgenannten ist die Signalübertragung, die hier aus numerischen Funktionen besteht. Schwellwerte legen fest ob eine durch eine Eingangsfunktion ermittelte Zahl zur Aktivierung oder Hemmung des Konzepts und angeschlossener Konzepte führen soll. Aber die Verfahren konnten nur auf einen kleinen Teil der Sprache getestet werden. Entscheidender Fortschritt in der WSD wurde durch die schnelle Weiterentwicklung der Computertechnologie erzielt. Durch Speicherung und Verarbeitung von großen Datenmengen wurde die automatische Disambiguierung ohne aufwändige manuelle Vorverarbeitung möglich. Texte konnten nun "extern", basierend auf Thesauri (z.B. Roget's Thesaurus) oder auf Maschine Readable Dictionaries (MRDs, wie z.B. WordNet) oder "intern", durch den Text selbst oder durch andere vorhandene Textauszüge (Korpora) disambiguiert werden. Hauptvor- und Nachteil der externen Quellen, insbesondere der Thesauri ist die Menge der Relation, die zu einem Wort gefunden werden können. So erleichtern Hierarchien, Taxonomien und auch eine große Menge von Synonymen die Einordnung des Wortes. Jedoch ist die manuelle Erstellung dieser Relationsangaben aufwändig und bei einem Teil der Disambiguierung, wie z.B. in der maschinellen Übersetzung, zu detailliert, da in der Zielsprache die Mehrdeutigkeit ebenso existieren kann. So konzentriert sich die aktuelle Entwicklung darauf große Textkorpora automatisch auszuzeichnen, abhängig vom micro-Kontext. Dazu werden "bag auf words", d.h. eine kleine Menge von dem Zielwort

umgebende Wörter zur Bedeutungsauflösung verwendet. Das heißt ein wichtiges Werkzeug der heutigen Verfahren der WSD sind Kollokationen. Im nächsten Kapitel wird nun ein Verfahren, das sich genau auf dieses Werkzeug bezieht, detaillierter vorgestellt.

2.2 Disambiguierungsansatz nach dem Algorithmus von David Yarowsky

Disambiguierung kann, wie schon in den Kapiteln vorher erwähnt, durch Menschen kontrolliert werden (supervised) oder völlig automatisch (unsupervised) ablaufen. Ziel der Forschung ist natürlich eine Automatisierung der WSD, jedoch kann dies eine sehr teure Angelegenheit werden und meist liefern supervised Modelle einfachere Verfahren und bessere Ergebnisse.

David Yarowsky präsentiert einen unsupervised Ansatz, der auf einem supervised Algorithmus basiert. Yarowsky benutzt in seinem Algorithmus zwei Bestandteile :

- **One sense per collocation:** Benachbarte Wörter (Kollokationen) bieten eine gute Möglichkeit den Sinn eines mehrdeutigen Wortes zu definieren
- **One sense per discourse:** Der Sinn eines mehrdeutigen Wortes bleibt bei mehrmaligen Auftreten innerhalb eines Dokumentes meist gleich.

Anhand einer Trainingsmenge bestehend aus über 37.000 Beispielen fand Yarowsky zum Einen Kollokationen, die er durch ein Bewertungssystem sortieren konnte, und zum Anderen bekräftigte er seine These des "One sense per discourse", denn wenn ein Wort in seinen Beispielen mehr als einmal in einem Dokument vorkam, dann wurde es zu 99,8 Prozent dem selben Sinn zugeordnet. Der Algorithmus besteht nun aus fünf Schritten. Zunächst werden alle Ambiguitäten mit ihrem Kontext aufgelistet. Dann werden durch die bewerteten Kollokationen in Schritt 2 und 3 Ambiguitäten nach und nach aufgelöst. In Schritt 4 nutzt Yarowsky "One sense per discourse" und löst dadurch eine weitere Menge von Mehrdeutigkeiten auf. Im letzten Schritt wird das ganze wiederholt, bis alle Mehrdeutigkeiten identifiziert wurden. Der Ansatz von Yarowsky hat eine Genauigkeit von ca. 96 Prozent, jedoch zeigt der Ansatz auch, dass WSD eine langatmige Angelegenheit sein kann, denn solch gute Ergebnisse, die hier erzielt werden, wurden nur durch handgeschriebene Beispiel erreicht. Dieser Algorithmus sollte einen Ansatz der WSD aufzeigen bevor wir nun zu unseren erarbeiteten Ansätzen kommen.

3 Implementierung

3.1 Einleitung

Nachdem wir zu Anfang des Praktikums begannen, uns Gedanken über mögliche Verfahren der Disambiguierung

zu machen, konnten wir zunächst nicht erahnen welcher Aufwand und welche Probleme auf uns zu kommen würden. Wir stellten uns die Frage, wie der Mensch beim Betrachten eines Textes eine Mehrdeutigkeit erkennt und auflöst und kamen zu dem Schluß dass beim Lesen ein Satz vielleicht mehr als einmal gelesen werden muss, um die Ambiguität eines Wortes oder eines ganzen Satzes zu erkennen und zu bestimmen. Aber im Gegensatz dazu war eine der Vorgaben unserer Aufgabe eine inkrementelle Betrachtung des vorliegenden Textes, Wort für Wort, ohne wiederholtes Einlesen eines Satzes. Zur Auflösung von Mehrdeutigkeiten stehen somit lediglich die linken Nachbarn des aktuellen Wortes zur Verfügung, wobei hier der Schwerpunkt von uns auf 5 Wörter festgelegt wurde. Das Ergebnis der Disambiguierung eines Wortes wird im weiteren Verlauf nicht verändert, eine einmal getroffene Entscheidung bleibt bestehen, wobei Entscheidungen sich auf nachfolgende Wörter auswirken können. Also im Gegensatz zum Menschen wird in unserem Programm kein Rückschritt im Satz möglich sein. Das Studieren von verschiedenen Verfahren, wie beim zuvor vorgestellte Ansatz von David Yarowsky, zeigte uns jedoch auf, welche langjährige Vorarbeit (zum Beispiel das Eintippen von mehreren Tausend Beispielen) nötig ist, um überhaupt effektive Ergebnisse beim Auflösen der Mehrdeutigkeit zu erzielen. Zur Vorbereitung unserer Ansätze stellten wir uns nun die Frage, welchen Weg der Disambiguierung wir einschlagen wollen und welche Vorarbeit bei diesen Ansätzen von Nöten sein würde. Doch bevor wir überhaupt einen Weg einschlagen konnten, warf schon die Analyse eines vorliegenden Textes einige Fragen auf. Welche Formen der Ambiguitäten wollen wir auflösen oder besser gesagt welche Formen sind für uns auflösbar? Hierbei wurde uns schnell klar, dass wir uns auf die lexikalische Mehrdeutigkeit, die nur ein Wort betrifft, beschränken, da es uns in der Kürze der Zeit unmöglich erschien einen Ansatz zur Auflösung von syntaktischer oder semantischer Ambiguitäten zu präsentieren. Es war nun wichtig zu entscheiden, welche Informationen wir zur Disambiguierung benötigen und welche Dinge wir vernachlässigen können. Es war unserer Meinung nach wichtig, die einzelnen Wörter eines Textes auf ihre Grundform zu reduzieren und ihre grammatikalische Zugehörigkeit zu bestimmen. Eine weitere wichtige Information zur Disambiguierung stellte für uns, auch inspiriert vom Verfahren von David Yarowsky, Kollokationen dar und wir entschieden uns den linken und den rechten Nachbarn eines jeden Wortes als Information zu speichern. Ausserdem sind natürlich Synonyme entscheidend für Verfahren der Disambiguierung. Nachdem wir nun wussten, welche Informationen für ein effektives Verfahren ausdrucksstark sein könnten, mussten wir nun die Frage klären, woher wir diese Infor-

mationen, wie Grundform, Grammatik oder Synonyme, erhalten. Für uns zeigten sich zwei Alternativen auf: erstens einen eigenen Thesaurus zu erstellen, in dem wir die Wörter eines Textes mit den genannten Informationen einspeichern oder zweitens eine externe Quelle für diese Informationen zu finden und in unsere Programme einzubauen. Der erste Punkt erschien uns in der kurzen Zeit nicht zu realisieren und zu umfangreich, um wirklich gute Ergebnisse zu erzielen. Ausserdem wollten wir den Versuch unternehmen, die Analyse auf beliebige Texte durchführbar zu machen. Zwar sah die Aufgabenstellung eine Eingrenzung auf eine bestimmte Domain vor, allerdings entwickelte sich im Verlauf der Recherchen und der Realisierung der Ehrgeiz, die Implementierung allgemeingültig zu erzeugen. Und da wir bei unserer Recherche auf eine externe Quelle stiessen, die uns genau die Informationen liefert und sich ausserdem als Schnittstelle in unsere Programme einbinden läßt, wurde dieser Ehrgeiz noch weiter gestärkt: das Projekt Wortschatz der Uni Leipzig.

3.2 Wortschatz der Uni Leipzig

Als Datenbasis für die gesamte Analyse eines Wortes wurde der Datenbestand des Projektes Wortschatz der Universität Leipzig gewählt (<http://wortschatz.uni-leipzig.de>). Dieses Projekt ist eine Sammlung von Wortformen der deutschen Sprache. Es liegen bisher ca. 2,5 Millionen Wortformen vor, die zum größten Teil aus Zeitungstexten, aber auch aus Fachtexten oder speziellen Wortlisten stammen. Bei der Abfrage eines Wortes erhält man sortiert nach ihrer Häufigkeit sämtliche relevante Informationen, welche wie folgt aufgelistet werden:

- Grundform
- Grammatikalische Zuordnung des Wortes
- Signifikante linke Nachbarn
- Signifikante rechte Nachbarn
- Synonyme
- Wortkategorien
- Beispielsätze
- Kookkurenzen
- Absolute Häufigkeit des Wortes
- Relative Häufigkeit des Wortes im Vergleich zum Wort "der"
- Assoziationsnetze (vergleiche dazu Abbildung 1)
- etc.

Der Zugriff auf den Datenbestand des Projektes Wortschatz erfolgt über den von der Universität Leipzig zur Verfügung gestellten Webservice. Darunter versteht man eine Softwareanwendung, welche auf einem Webserver ausgeführt wird und eine definierte Schnittstelle besitzt, welche eine Interaktion auf Basis von XML¹-Nachrichten ermöglicht. Hierdurch kann ein Zugriff unterschiedlicher Programme, welche mit unterschiedlichen Programmiersprachen implementiert wurden, erfolgen. Alle im Bestand von Wortschatz gespeicherten Informationen kann man also durch den Webservice über eine Abfrage beziehen. Die Limit-Angabe, welche die Anzahl der gelieferten Werte begrenzt, wurde hierbei auf den Wert 20 eingestellt. Dies bedeutet, dass beispielsweise eine Abfrage der Synonyme des Wortes maximal 20 Synonyme, entsprechend ihrer Häufigkeiten, liefert. Um die Abhängigkeit einer Internetverbindung zu vermeiden (jeder Zugriff auf den Webservice setzt natürlich eine aktive Internetverbindung voraus) und um eine bessere Performance zu gewährleisten, werden sämtliche vom Webservice bezogenen Werte in einem lokalen Datenbestand geführt. Als Datenstruktur dient hier abermals XML, wobei für jedes Wort ein eigener Knoten (<item>) erstellt wird und die einzelnen Werte in entsprechenden Unterknoten gespeichert werden. Diese Verwaltung verhindert auch doppeltes Beziehen von Informationen zu einem Wort, da bei Abfragen zunächst mit der lokalen Datenbasis abgeglichen wird. Somit entsteht eine auf XML basierende Datenstruktur. Innerhalb dieser Struktur wird in unterschiedlichen Knoten der gesamte Text, in Wörter geteilt, angereichert mit den entsprechend benötigten Informationen (Grundform - base, grammatikalische Zuordnung - grammar, signifikante linke/rechte Nachbarn - left/right, Kookurenzen - coocurrences, Häufigkeit - frequencies, Synonyme - synonyms und Wortkategorien - categories) abgelegt und nach vollständiger Disambiguierung mit den Zusatzinformationen ausgegeben. Somit entsteht nach und nach ein Wörterbuch (dictionary.xml).

Bei der Verwendung des Webservice sind, neben der nicht immer stabilen Internetverbindung und nicht immer funktionierenden Webseite der Uni Leipzig, besonders bei der Implementierung der für den Zugriff auf den Webservice benötigten Routinen Probleme aufgetreten. Es stellte sich heraus, dass für die Autorisierung der Anwendung eine Kombination aus Username und Passwort benötigt wird. Die Recherche ergab, dass für einen freien Zugriff hier als Username *anonymous* und als Passwort ebenfalls *anonymous* anzugeben ist. Die Übergabe dieser Autorisierungsdaten erfolgt BASE64-

¹Extensible Markup Language: Es ermöglicht uns, unterschiedliche maschinen- und menschenlesbare Informationen in einer Baumstruktur darzustellen und eignet sich somit sehr gut zum Datenaustausch mit einem Webservice.

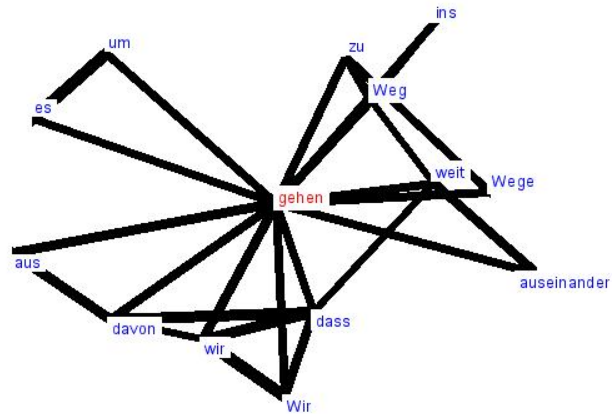


Figure 1: Assoziationsnetz der signifikanten linken Nachbarn des Wortes "gehen"

```

- <dictionary>
+ <item></item>
- <item>
  <word>gehen</word>
  - <baseform>
    - <baseitem>
      <base>gehen</base>
      <grammar>V</grammar>
    </baseitem>
  </baseform>
+ <left></left>
+ <right></right>
+ <cooccurrences></cooccurrences>
+ <frequencies></frequencies>
+ <synonyms></synonyms>
+ <categories></categories>
</item>

```

Figure 2: Auszug aus der Datei dictionary.xml am Beispiel "gehen"

Verschlüsselt. Leider stellte sich im Verlauf der Implementierung der Webservice-Zugriffe heraus, dass keiner der Zugriffe die einem Wort entsprechenden Sachgebiete übermittelt. Alle anderen Zugriffe verliefen ohne weitere Probleme und die Informationen, wie grammatikalische Zuordnung oder Beispielsätze liessen sich ohne Probleme abrufen. Somit mussten wir einen anderen Weg finden, die zu einem Wort zugehörigen Sachgebiete von Wortschatz abzurufen, da sie für einen unserer Disambiguierungsansätze eine entscheidende Information darstellen. Nun bestand hier die Schwierigkeit darin, direkt die Ergebnisse der Website auszuwerten. Unser Programm muss also statt den Webservice zu nutzen, direkt die Website `wortschatz.uni-leipzig.de` aufrufen, dann das Textfeld der Webseite mit dem aktuellen Wort füllen, den Hacken für die Beachtung der Groß-/Kleinschreibung setzen und dann noch auf den Button klicken und die Suche bestätigen. Das Ergebnis ist eine Übersicht der unterschiedlichen Werte des Wortes (Grundform, Kookurenzen, Synonyme und und und ...) und hier musste jetzt ein Parser geschrieben werden, welcher nur die relevanten Sachgebiete aus der Fülle aller Informationen rausfiltert und abspeichert. Alle diese Maßnahmen wurden nötig, da leider keine Hilfe zum Abruf der Sachgebiete über den Webservice von Wortschatz zur Verfügung standen. Im Programm selbst wird dies in einem kleinen Fenster oben rechts dargestellt und somit lässt sich der gesamte Verlauf verfolgen (siehe Abbildung 2). Ein weiteres Problem, dass beim Verwenden des Webservice auftrat, war das Speichern der Knoten. Abseits der gängigen Vorgehensweise, welche vorsieht, dass zu übergebende Informationen in einem speziell für diesen Zweck zur Verfügung gestellten Knoten übermittelt werden, muss beim Zugriff auf den Webservice die Klassifizierung der Übergabe im gleichlautenden Knoten erfolgen.

- gängige Vorgehensweise `<wort>Burg</wort>`
- Wortschatz-Webservice
`<dataRow>Wort</dataRow>`
`<dataRow>Burg</dataRow>`

3.3 Preprocessing

Um einen beliebigen Text analysieren zu können, muss der zu disambiguierende Text zunächst in einem Editor als txt-Datei gespeichert werden. Im Programm selbst ist es dann möglich die erstellte Datei auszuwählen. Vor der eigentlichen Disambiguierung wird zunächst das aktuelle Offline-Wörterbuch (`Dictionary.xml`) auf Vollständigkeit geprüft. Dazu wird der zur Disambiguierung stehende Text in Wörter gesplittet und jedes gefundene Wort wird im Wörterbuch nachgeschlagen. Die Routine zur Prüfung auf Vollständigkeit des Wörterbuches legt, im Falle eines neuen Wortes, dieses entsprechend

im Wörterbuch an. Zusätzlich werden die entsprechend benötigten Informationen, wie Synonyme, Sachgebiete, Grundform usw. wie zuvor beschrieben (entweder über den Webservice oder direkt über die Seite) abgerufen und hinzugefügt. Welche Daten abgerufen werden, kann man über das Konfigurationsmenü direkt im Programm einstellen. Also man kann bevor das Programm über einen beliebigen Text läuft, entscheiden welche Informationen von Wortschatz zu jedem Wort abgerufen werden. Je nach Bedarf kann somit die Größe des Wörterbuches gesteuert werden und für den Benutzer irrelevante Informationen nicht zum Abruf freigegeben werden. Ausserdem lässt sich im Programm über zwei Textfelder die Delimiter und die Anzahl der Predecessors festlegen. Der Delimiter setzt die Trennzeichen für die Extraktion von Wörtern aus dem Text. Dies ist notwendig, da in einem Text auch andere Trennzeichen wie Bindestrich, Apostroph oder andere Trennzeichen auftreten können. Die Zahl der Predecessors legt die Anzahl der Vorgänger fest, d.h. die linken Nachbarn, die für das aktuelle Wort betrachtet werden sollen. Da unsere Verfahren inkrementell arbeiten soll, dürfen wir nur die linken Nachbarn in diesem Radius berücksichtigen. Standardmäßig stehen die Satzzeichen auf `„,:!?”` und die Anzahl der Predecessors auf fünf. Die Trennung der Textzerlegung und der eigentlichen Algorithmen erlaubt es uns, verschiedene Ansätze auszuprobieren und auch spätere Ideen noch in das Programm einzubauen. Die Grundlage für die Algorithmen wird also im Preprocessing gelegt, so dass mit einem beliebigen Algorithmus darauf aufgebaut werden kann.

3.4 Algorithmus der Disambiguierung

Unser Algorithmus basiert auf dem Vergleich von Sachgebieten (natürlich auch abgerufen über Wortschatz). Von Wortschatz eingesetzte Sachgebiete sind zum Beispiel Medizin, Wissenschaft, Ökonomie oder Literarische Motive Stoffe Gestalten. Die Idee hinter dem Algorithmus basiert auf der Annahme, dass jedem Wort unterschiedliche Sachgebiete zugeordnet werden können, welche gleichzeitig die Bedeutung des Wortes kategorisieren. Als Beispiel sei hier das Wort "Schloß" genannt, welches entweder dem Sachgebiet "Gebäude" oder aber "Technik" (Schließvorrichtung) zugeordnet werden kann. So kann jedes einzelne zu disambiguierende Wort einkategorisiert werden. Jedoch basiert der Algorithmus nicht nur auf den Kategorien des entsprechenden Wortes, denn das w"urde immer dieselbe Kategorie für mehrdeutige Worte ergeben, sondern vielmehr müssen natürlich auch die im Kontext stehenden Nachbarn des aktuellen Wortes betrachtet werden und deren Kategorien mit einbezogen werden. Da eine unserer Anforderungen, wie schon in den Kapiteln zuvor erwähnt, eine inkrementelle Betrachtung

vorsah, durften zur Analyse der Bedeutung lediglich die linken Nachbarn betrachtet werden. Als weitere Gewichtung sollen auch die Sachgebiete der Synonyme des aktuellen Wortes betrachtet werden. Die Gewichtung der einzelnen Berechnungsfaktoren (aktuelles Wort, Synonyme, linke Nachbarn) soll variabel gehalten werden, um das optimale Ergebnis durch Vergabe von entsprechenden Gewichtungen zu erreichen, d.h. im Programm selbst soll einstellbar sein auf welches der drei Kriterien man am Meisten Wert legt. Dies macht es möglich sich auf die Anforderungen von verschiedenen Texten einzustellen und auszutesten, welche Einstellungen für einen Text die besten Ergebnisse liefern (mehr dazu in Kapitel 3.3). Nach der Addition der Gewichte eines jeden in Betracht kommenden Sachgebietes soll jenes Sachgebiet ermittelt werden, welches die höchste Wertigkeit aufweist. Wie genau die einzelnen Wertigkeiten zu Stande kommen, wird im nächsten Kapitel an einem Beispiel dargestellt. Der Algorithmus wird, wie schon in der Einleitung erwähnt, als supervised- und als unsupervised-Verfahren implementiert. Während die unsupervised-Variante die errechnete Kategorisierung vollständig übernimmt, wird die supervised-Variante den beschriebenen Algorithmus lediglich zur Ermittlung des optimalen Sachgebietes wählen. Hier tritt im Anschluss an die Berechnung der Gewichte der User in Aktion und entscheidet, ob der vom Algorithmus errechnete Wert seiner Interpretation der Bedeutung entspricht. Als Hilfestellung wird ihm der Kontext bestehend aus den fünf linken Nachbarn des aktuellen Wortes dargestellt, jedoch keine rechten Nachbarn, da auch dieses Verfahren komplett inkrementell ablaufen soll. Des Weiteren kann der User seine Entscheidung ebenfalls klassifizieren, so dass bei Unsicherheit des Users seine Entscheidung mit einer kleineren Gewichtung angenommen wird und wenn sich der User bei seiner Interpretation sicher ist mit einer hohen Gewichtung angenommen wird.

3.5 Anwendung des Algorithmus

Um die unterschiedlichen Gewichte und zu betrachtenden Sachgebiete konfigurieren zu können, stellt die Programmoberfläche verschiedene Entscheidungsabfragen und Einstellungsmöglichkeiten bereit. Abbildung 3 soll diese Möglichkeiten illustrieren. Zunächst hat der User die Möglichkeit, festzulegen, wie viele linke Nachbarn des aktuellen Wortes betrachtet werden sollen. Hierbei dürfen und können nicht mehr linke Nachbarn betrachtet werden, als das Preprocessing ermittelt und gespeichert hat. Das Programm kontrolliert diese Einstellung und erlaubt hier keine falschen Angaben. Dazu enthält das Programm zwei weitere Auswahlkriterien. Erstens kann der User entscheiden, ob Synonyme in die Betrachtung einbezogen werden sollen. Der zweite Punkt stellte sich nach einigen Durchläufen be-

liebigere Texte als Notwendig dar. Einige Tests zeigten nämlich auf, dass die Betrachtung aller Wörter, also auch solcher, welche keine eindeutige grammatikalische Zuordnung haben, zu Fehlergebnissen führt. Somit kann man mit einer zusätzlichen Einstellung festlegen, dass nur Wörter mit einer eindeutigen grammatikalischen Zuordnung in Betracht gezogen werden. Die Gewichtung der drei Analysebereiche (Sachgebiete des aktuellen Wortes, der Synonyme und der linken Nachbarn) kann durch Vergabe von Werten zwischen 0 und 20 erfolgen. Hierbei schließt eine Einstellung von 0 bei einem Bereich diesen aus der Betrachtung aus. Eine weitere Schwierigkeit, die sich bei verschiedenen Tests herausstellte ist, dass überproportional viele Wörter eine Zuordnung in das Sachgebiet "Nachname" haben. Aus diesem Grund wurde, in Anbetracht der universell beziehbaren Daten über den Wortschatz-Webservice, zusätzlich eine Möglichkeit implementiert, bestimmte Sachgebiete aus der Betrachtung auszuschließen. Jedoch müssen wir nun berücksichtigen, dass beispielsweise das Ausschließen des Sachgebietes "Nachname" zu einer Fehlberechnung bei Wörtern führt, welche tatsächlich einen Nachnamen darstellen. Als Standard sind hier die Sachgebiete "Nachname" und "Motive" eingetragen, jedoch lassen sich über das Textfeld und den Button "Add" andere Sachgebiete aus der Betrachtung ausschließen.

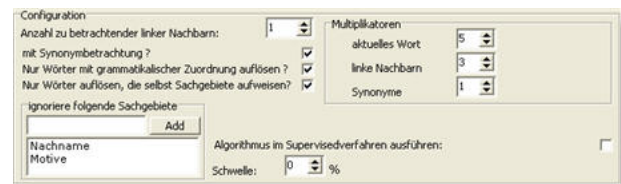


Figure 3: Auswahlmöglichkeiten der Programmoberfläche

Der wohl wichtigste und für den Algorithmus entscheidende Punkt stellt die Einstellung der Multiplikatoren dar. Wie schon im Kapitel zuvor beschrieben werden den Sachgebieten Wertigkeiten zugeordnet. Das Sachgebiet mit der höchsten Wertigkeit wird dann dem betrachteten Wort zugeordnet. Wir gehen von den drei Arten aus, die mit in die Berechnung einbezogen werden: Sachgebiete des aktuellen Wortes, der Synonyme und der linken Nachbarn. Diesen drei Arten wird im Programm durch den User ein Multiplikator zugewiesen. Als Standard stehen die Multiplikatoren auf 5 (aktuelles Wort), 3 (linke Nachbarn) und 1 (Synonyme), d.h. die Sachgebiete des aktuellen Wortes werden am Meisten in Betrachtung gezogen. Zuerst geht das Programm die Sachgebiete des aktuellen Wortes durch und vergleicht ob das Sachgebiet, welches es gerade betrachtet, schon in der Liste ist. Wenn nicht, dann wird es eingetragen

und als Gewicht wird der vom User gewählte Multiplikator eingetragen. Wenn ja, dann wird nur das eingestellte Gewicht zum bisherigen Gewicht hinzuaddiert. Diese Berechnung wird nacheinander mit den Sachgebieten des aktuellen Wortes, der linken Nachbarn und zum Schluß mit den Synonymen durchgeführt. So wird den einzelnen Gebieten eine Wertigkeit zugeordnet. Das unsupervised-Verfahren übernimmt diese Wertigkeiten direkt und soll somit die vorliegende Ambiguität auflösen. Eine letzte Auswahlmöglichkeit stellt ein Haken dar, der nur Wörter auflösen lässt, die selbst Sachgebiete aufweisen. Ist diese Einstellung gesetzt, so werden lediglich Wörter der Disambiguierung unterzogen, welche mindestens einem Sachgebiet zugeordnet sind. Hierdurch wird verhindert, dass Wörter, welche direkte Nachbarn rechte Nachbarn eines Wortes sind bei der Disambiguierung die Bedeutung des linken Nachbarn zugeordnet bekommen, obwohl sie im Grunde genommen eindeutig sind. Die war häufig bei Verben zu beobachten, welche direkt nach einem disambiguierten Wort auftraten.

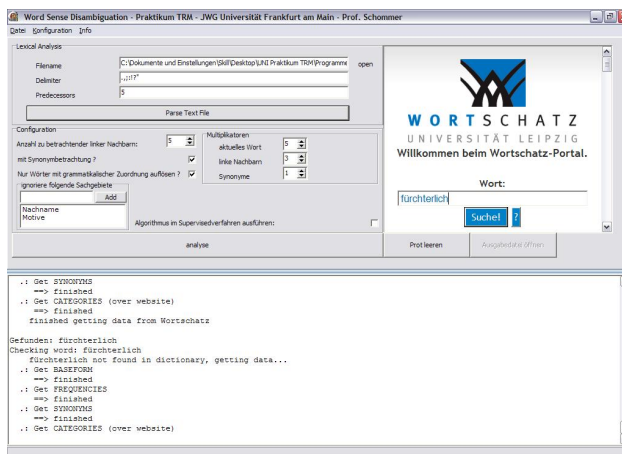


Figure 4: Programmoberfläche, mit dem Wortschatz-Thesaurus der Universität Leipzig

Für das supervised Verfahren lässt sich nun noch eine Schwelle in Prozent einstellen, welche bestimmt, ob im supervisedmodus eine Abfrage stattfindet. Ist dieser Schwellwert erreicht oder überschritten, so wird das errechnete Sachgebiet des aktuellen Wortes als eindeutig angenommen. Die folgenden Sätze wurden mit den Kategorien, welche mit der berechneten Häufigkeit hinter dem Wort in Klammer stehen, disambiguiert. Dabei wurden die $d = 5$ Vorgänger betrachtet. Die Multiplikatoren waren auf $g_W = 5$ für das aktuelle Wort, $g_V = 3$ für die linken Nachbarn und $g_S = 1$ für die zum Wort gelieferten Synonyme eingestellt. Unbetrachtet blieben die Kategorien "Vorname,Nachname,Motive", da diese bei Wortschatz zu allen Wörter zugeordnet, also nicht aussagekräftig sind. Das *Schloss* [Architektur,5] *liegt*

[Architektur,3] in der *Nähe* [Technik,5] der *Stadt* [Anthropogeographie,15] Er *setzte* [Anthropogeographie,3] sich auf die *Bank* [Möbel,10] und unternahm was *schönes* [Lexikologie,5] Er *hob* [Lexikologie,3] *Geld* [Ökonomie,15] von der *Bank* [Möbel,10] ab und brachte es in das *Haus* [Architektur,31].

3.6 Der supervised-Modus

Ein letzter Punkt der Programmoberfläche lässt dem User die Entscheidung, ob das supervised- oder das unsupervised-Verfahren angewendet werden soll. Wenn der Algorithmus im Supervised-Modus ausgeführt wird so wird nach jedem analysierten Wort, welches einer Disambiguierung unterzogen wurde, das Ergebnis mit entsprechenden Auswahlmöglichkeiten dem User eingeblendet. Dieser kann dann seine Entscheidung anhand der vorgelegten Sachgebiete mit zugeordneter Wertigkeit treffen. Dem User wird bei seiner Entscheidung jedoch nur der Kontext gezeigt mit den letzten linken Nachbarn (inkrementelle Betrachtung). Unser Ziel war es, Entscheidungen des Users in die spätere Auflösung der Ambiguität einfließen zu lassen, jedoch stellten sich hier beim Implementieren einige Probleme heraus. Da das Programm immer das Sachgebiet als errechnetes heranzieht, welches als erstes mit dem maximalen Wert in der Liste auftaucht, kann es sein, dass zwar das eigentlich richtige Sachgebiet mit der gleichen Gewichtung vorkommt, aber nicht genommen wird, weil ein anderes vor diesem steht (mit dem gleichen Gewicht). Außerdem war es uns nicht möglich das Problem von Falschaussagen zu vermeiden, wenn ein Wort (z.B. Bank) zwei Mal im Text vorkommt und dabei zwei verschiedene Bedeutungen hat.

3.7 Ausgabe der Ergebnisse

Da die Speicherung der ausgewerteten Daten in einer XML-Datenstruktur vorgenommen wird, wird für die eigentliche Ausgabe des Ergebnistextes diese Datei herangezogen. Mittels einer XSL-Transformation wird die XML-Struktur in einen Fließtext umgewandelt, wobei Wörter, welcher der Disambiguierung unterzogen wurden, als Link erscheinen. Die Positionierung des Maus-cursors über solch einem Wort führt zur Einblendung einer Informationsbox, welche die berechnete Bedeutung des aktuellen Wortes ausgibt. Diese Informationsbox zeigt dann das Sachgebiet an, welches dem Wort zugeordnet wurde.

3.8 Weitere mögliche Algorithmen

Der aktuell gelesene Text ist $T_a = w_1 \dots w_i \dots w_n$, das aktuelle Wort w_n sowie die d Vorgänger w_{n-d}, \dots, w_{n-1} haben die von Wortschatz gelieferten k unterschiedlichen Kategorien C_{n_1}, \dots, C_{n_k} . Für die Kategorien werden Gewichte vorgegeben, und zwar g_W für Kategorien des

aktuellen Wortes und g_V für Kategorien der Vorgänger. Insgesamt erhalten wir also die aktuellen Häufigkeiten $H_a(C_{n_1}), \dots, H_a(C_{n_k})$. Jedes der $H_a(C_{n_i})$ kann dabei Werte von g_W bzw. g_V bis maximal $g_W + d \cdot g_V$ annehmen. Bis hier hin entspricht der Ansatz unserem vorgestellten Algorithmus. Nun wollen wir weitere mögliche Algorithmen vorstellen, die aufgrund von Zeitmangel leider nur theoretisch entstanden sind und noch nicht ausführlich getestet werden konnten. Betrachtet man längere Texte, könnte man Yarowskys zweiter These "One sense per discourse" folgend annehmen, dass sich die Bedeutung eines Wortes kaum ändert auch wenn es mehrmals im Text auftritt. Zur Umsetzung dieser Annahme müssten wir den Algorithmus ermöglichen, für alle Wörter, die sich unterscheiden, die letzten Häufigkeiten zu speichern. Anstatt der aktuellen Häufigkeit würden wir dann mit den summierten Häufigkeiten $H_S(C_{n_1}), \dots, H_S(C_{n_k})$ die Disambiguierung des Wortes durchführen. Ist das aktuelle Wort w_n zuletzt an i -ter Position, also als w_i aufgetreten, dann wären die aktuellen Häufigkeiten zu bestimmen als $H_S(C_{n_1}) = H_a(C_{n_1}) + H_a(C_{i_1}), \dots, H_S(C_{n_k}) = H_a(C_{n_k}) + H_a(C_{i_k})$. Das Ergebnis dieser Erweiterung wäre, wenn man das Gewicht g_V der Vorgänger niedrig und das Gewicht g_W der Wörter relativ hoch wählt, das die bestimmte Kategorie sich kaum mehr ändern sollte. Als nächstes betrachten wir nun die Vorgänger genauer. Wenn es nach Gale, Church und auch Yarowsky geht sind Vorgänger, die näher am Wort sind wichtiger für die Disambiguierung. Im Algorithmus müsste man die Abstände der Vorgänger zum aktuellen Wort gewichten und für die Häufigkeiten $H_a(C_{n_i})$ hätte man zum Aufsummieren anstatt der identischen Gewichte g_V die Gewichte $g_{V_d} = \frac{g_V}{d}$ für den d letzten Vorgängen bis hin zu $g_{V_1} = \frac{g_V}{1}$ für den direkten Vorgänger. Die dritte Modifizierung unseres Algorithmus ist orientiert an der Tatsache, dass im Gegensatz zu Yarowskys zweiter These, in einem langen Text durchaus ein Wort seine Bedeu-

tung ein wenig wandeln kann. So hat das Wort "Interesse" sowohl eine rein finanzielle (geschäftliches Interesse), als auch eine soziale Bedeutung (Sorge, Neugier) haben. Jedoch können durchaus beide Bedeutungen in einem Text auftreten. Um für ein Wort mehrere Bedeutungen zuzulassen, müsste der Algorithmus alte Bedeutungen, die wir in der ersten Erweiterung hinzugefügt haben wieder "vergessen". Das bedeutet, dass der letzte Häufigkeitswert einer Kategorie für ein Wort nach gewisser Nicht-Aktualisierung geringer wird. Für die Berechnung der Summen Häufigkeit ergibt sich dadurch: $H_S(C_{n_1}) = H_a(C_{n_1}) + t \cdot H_a(C_{i_1}), \dots, H_S(C_{n_k}) = H_a(C_{n_k}) + t \cdot H_a(C_{i_k})$ wobei t eine zeitliche Gewichtung sein muss, also $t = e^{i-n}$.

4 Fazit

Das vorangegangene Kapitel zeigt, dass neben unserem Hauptalgorithmus auch noch andere Ansätze hätten verfolgt werden können. Nachdem uns jedoch schon sehr früh der Ehrgeiz gepackt hatte, das Verfahren auf beliebige Texte anwenden zu können, war der Zeitrahmen leider zu knapp bemessen, um weitere Algorithmen zu implementieren und ausführlich zu testen. Das Testen ist sicher ein weiterer Punkt, den man ansprechen sollte, denn längeres Testen an weiteren verschiedenen Texten wird die Einstellungen, z.B. der Multiplikatoren, weiter ausreifen und das Verfahren effektiver machen. Ein weiterer Verbesserungsvorschlag wäre Sätze eigenständig zu betrachten und nicht satzübergreifend nach linken Nachbarn zu suchen. Jedoch kam uns diese Idee leider erst zu spät. Desweiteren ist sicher der Dictionary ein Kritikpunkt, weil wir keinen Einfluss auf die Daten von Wortschatz hatten und sich Ergebnisse vielleicht im Laufe der Zeit ändern könnten und Webservices vielleicht nicht mehr angeboten werden. Somit wäre mehr Unabhängigkeit sicher vorteilhaft. Jedoch wollen wir hier zum Abschluß ein großes Lob und Dank an das Team von Wortschatz der Uni Leipzig richten, da ohne ihre Arbeit unser Ansatz nur schwer möglich gewesen wäre und sie uns meist bei Fragen immer zur Seite standen. Wir hoffen, dass wir mit unserem Nutzen des Web-service, oft über viele Stunden, nicht die Leitungen lahm gelegt haben. Zu unseren Ergebnissen ist zu sagen, dass sicher noch einiges zu verbessern ist und mehr Tests erforderlich sind, um für beliebige Texte bessere Ergebnisse zu erzielen. Jedoch hätten wir zu Beginn des Praktikums nicht erwartet, dass die Ergebnisse so gut bei beliebigen Texten aussehen würden. Besonders an unserem Beispielsatz, der auch hier in der Doku zu finden ist, sieht man sehr gut, dass durch Probieren nach und nach immer bessere Ergebnisse zu verzeichnen waren. In Zukunft könnte man, auf unseren Ergebnissen aufbauend, sicher noch weitere Ansätze verfolgen. Zum Einen könnte man statt der Sachgebiete die Synonyme zu betrachten, um

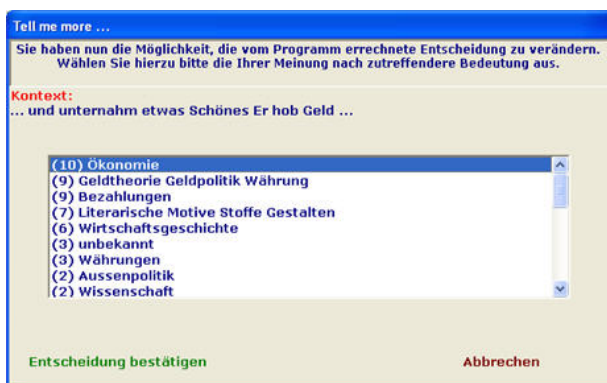


Figure 5: Auswahlfenster der Supervised Methode

vielleicht bessere Ergebnisse zu erzielen. Zum Anderen wäre es interessant, das Gesamtsachgebiet eines Textes zu bestimmen, um somit vielleicht eine Kategorisierung beliebiger Texte zu erreichen.

5 Acknowledgement

Die vorliegende Arbeit fand im Rahmen des Praktikums *Text Mining and Retrieval* im Sommersemester 2006 an der JW Goethe-Universität Frankfurt am Main unter der Leitung von Prof. Dr. Christoph Schommer² statt.

6 References

- [1] WikiLingua, Universität Trier, Fachbereich Computerlinguistik, http://www.uni-trier.de/uni/fb2/ldv/ldv_wiki (Stand 2006)
- [2] Wikipedia, <http://de.wikipedia.org/wiki/Ambiguität> (Stand 2006)
- [3] Forum zur Delphi-Programmierung, <http://www.delphipraxis.net> (Stand 2006)
- [4] Unsupervised Word Sense Disambiguation rivaling supervised Methods. http://www.cs.nyu.edu/courses/spring04/G22.2592-001/Waxmonsky_WSD.ppt (Stand 2006)
- [5] Wortschatz Uni Leipzig. <http://www.wortschatz.uni-leipzig.de> (Stand 2006).
- [6] Eneko Agirre, David Martinez. Knowledge Sources for Word Sense Disambiguation. <http://www.citebase.org/cgi-bin/citations?id=oai:arXiv.org:cs/0109030> (Stand 2001).
- [7] Marti A. Hearst. Noun homograph disambiguation using local context in large corpora. Proceedings of the 7th Annual Conference of the University of Waterloo Centre for the New Oxford English Dictionary, Oxford, UK (1991).
- [8] Jochen Leidner: Linksassoziative morphologische Analyse des Englischen mit stochastischer Disambiguierung. <http://www.linguistik.uni-erlangen.de/files/maleidne.pdf>.
- [9] Nancy Ide, Jean Veronis: Word Sense Disambiguation: The State of the Art, Computational Linguistics. Vassar College, Universite de Provence. 1998.
- [10] Rada Mihalcea, Ted Pedersen. Advances in Word Sense Disambiguation, Tutorial at AAAI-2005. 2005. <http://www.d.umn.edu/~tpederse/Tutorials/ADVANCES-IN-WSD-AAAI-2005.ppt>
- [11] David Yarowsky. Department of Computer and Information Science University of Pennsylvania, Philadelphia, PA 19104, USA. 1995.

²University of Luxembourg, Campus Kirchberg, Dept. of Computer Science and Communication, 6 Rue Coudenhove-Kalergi; 1359 Luxembourg, Luxembourg, Email: christoph.schommer@uni.lu