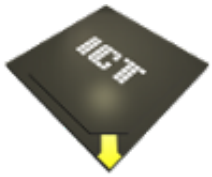
TUHH
Technische Universität Hamburg-Harburg



Institut für Eingebettete Systeme

Auflösung von sprachlichen Mehrdeutigkeiten für spezifische Domänen mittels Simulated Annealing

Bachelorarbeit

Hamburg, den 13. Oktober 2015

Autor:
Zainab Alsewan

Prüfer: Prof. Dr. Karl-Heinz Zimmermann
Zweitprüfer: Sallam Abualhaija

Erklärung:
Hiermit versichere ich, Zainab Alsewan, diese Arbeit im Rahmen der an der Technischen Universität Hamburg-Harburg üblichen Betreuung selbstständig angefertigt und keine anderen als die angegebenen Quellen und Hilfsmittel eingesetzt zu haben.

Zainab Alsewan

Hamburg, den 13. Oktober 2015

Abstract

Computer können Texte nicht interpretieren und verstehen. Dafür wird ein System benötigt, das den Computern dabei hilft, Texte zu verstehen. Dieses System wird Word Sense Disambiguation (WSD) genannt. Es stellt ein kombinatorisches Optimierungsproblem dar. Die Adaptation von Domänen in WSD Systeme sind neue Herausforderungen bei WSD. In dieser Arbeit erfolgt in Kapitel 1 eine Einführung in die Thematik. In Kapitel 2 wird „Word Sense Disambiguation als Optimierungsproblem“ vorgestellt. Zu dem wird eine vorherige Arbeit präsentiert. Anschließend werden die Eingabedaten des Systems wie WordNet und Dataset beschrieben. Den Hauptteil der Arbeit stellt Kapitel 4 dar. Hier werden die entwickelten Algorithmen von WSD mittels Simulated Annealing vorgestellt. Im Kapitel 5 werden die erstellten Algorithmen mit dem Korpus aus dem SemEval 2010 Task17 und dem Korpus aus dem SemEval 2007 Task 7 evaluiert. Anschließend werden die Ergebnisse präsentiert und analysiert. Am Schluss erfolgt die Zusammenfassung der Arbeit.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Häufig besitzen die Wörter in allen natürlichen Sprachen Mehrdeutigkeiten. Der Mensch kann durch seine Erfahrungen und sein Wissen die passende Bedeutung automatisch erkennen. Computer haben jedoch Schwierigkeiten, selber die richtige Bedeutung eines Wortes aus einem Kontext herauszufinden. Dieses Problem wird in der Computerlinguistik (CL) – oder linguistischen Datenverarbeitung (LDV) - als „Word Sense Disambiguation“ (WSD) bezeichnet. Im Deutschen es als „Auflösung sprachlicher Mehrdeutigkeit“ bezeichnet.

Unter Word Sense Disambiguation versteht man den Prozess, bei dem einem bestimmten Wort eine Bedeutung anhand eines Kontextes zugewiesen wird. In den meisten Fällen sind die möglichen Bedeutungen eines Wortes im Vorfeld bekannt und können maschinell nachgeschlagen werden. Word Sense Disambiguation spielt im Bereich der linguistischen Datenverarbeitung seit Jahren eine große Rolle in der Forschung. In letzter Zeit sieht man sinnvolle Einsatzmöglichkeiten von WSD in LDV, wie bei der maschinenunterstützer Übersetzung, der automatischen Informationsgewinnung und bei der Beantwortungen von Fragen durch Maschinen. WSD ist einer der bedeutendsten Bausteine in der LDV, damit Maschinen natürliche Sprachen verstehen können.

1.1 WSD Methoden

Es gibt unterschiedliche Ansätze für die Auflösung der Mehrdeutigkeit, wie überwachte, nicht überwachte und wissensbasierte WSD-Systeme.

Überwachte WSD Systeme werden mit Hilfe manuell richtig disambiguierter Beispiele innerhalb eines Kontextes trainiert. Dadurch werden bessere Ergebnisse als durch nicht überwachte Systeme erreicht. Aber diese Systeme haben den Nachteil, dass passende Trainingstexte für neu zu disambiguierende Korpora bereitgestellt werden müssen.

Nicht überwachte WSD Systeme brauchen keine Texte zum Trainieren. Somit entfällt der Nachteil, einen Trainingstext manuell zu generieren. Diese Technik verfolgt ein anderes Ziel als überwachte-Systeme und wissensbasierte Systeme. Bei dieser Technik werden keine Bedeutungen den Wörtern zugewiesen, sondern die Wörter werden in Gruppen geteilt. Diese werden im nächsten Schritt Bedeutungen zugeordnet [5].

Des Weiteren wird zwischen wissensbasierten (basiert auf Wissensquellen wie Wörterbuch etc.) und korpusbasierten (arm an Wissen, da Korpora nur begrenzt Wissen enthalten) Ansätzen unterschieden. Wissensbasierte Methoden sind nicht performant im Vergleich zu überwachten Systemen. Allerdings decken sie dank der Ausnutzung der großen Wissensquellen ein breites Spektrum ab.

1.2 Aufgabenstellung

Ein wichtiger Aspekt beim WSD ist die Größe des semantischen Raums, in dem das mehrdeutige Wort erscheint. Der Einfluss einer bestimmten Domain beim Auflösen der Mehrdeutigkeit kann man an sprachübergreifenden Beispielen gut illustrieren, wie sie in Maschinenübersetzungen erscheinen würden. So wird zum Beispiel das englische Wort „housing“ in einem sehr allgemeinen Kontext als „Wohnung“ übersetzt. Es wird jedoch im Ingenieurwesenbereich als Gehäuse übersetzt. Das Problem tritt auch bei anderen Wortarten auf. Zum Beispiel wird das Verb „warming up“ in einem allgemeinen Kontext als „erhitzen“, im Sportbereich jedoch als „aufwärmen“ ins Deutsche übersetzt [8]. Somit wird anhand der genannten Beispiele sehr deutlich, dass Domänen eine wichtige Rolle beim Disambiguieren spielen. Daher wird in dieser Arbeit ein System zur Auflösung der Mehrdeutigkeiten für spezifische Domänen mittels Simulated Annealing entwickelt.

Wenn man nur einige wenige Wörter aus einem Text disambiguieren möchte, so genügt ein Algorithmus, der Wort für Wort disambiguiert. Aber bei großen Korpora, bei denen viele Wörter zu disambiguieren sind, müssen Algorithmen mehrere Wörter gleichzeitig in machbarer Zeit auflösen können. Dieses wird durch Simulated Annealing ermöglicht.

2 Word Sense Disambiguation als Optimierungsproblem

In den vergangen Jahren entstand eine große Vielfalt an Forschungsarbeiten über Word Sense Disambiguation.

Im Folgenden wird eine WSD-Verfahrensweise, bei der einem Zielwort der Sinn anhand der Maximierung der Beziehung zwischen dem Zielwort und seinen Nachbarn zugewiesen wird, vorgestellt. Diese Methode wurde von Banerjee und Pedersen in [1] vorgestellt. Sie haben den Lesk [4] Algorithmus so angepasst, dass WordNet als Wissensquelle benutzt werden konnte.

Lesk Algorithmus disambiguiert ein aufzulösendes Wort, indem die Definition der Bedeutung des Zielwortes mit den Definitionen der Bedeutungen der Nachbarwörter verglichen wird. Die Bedeutung des Zielwortes, die die größte Schnittmenge mit den Bedeutungen der Nachbarwörter bildet, wird als die passende Bedeutung für das Zielwort in dem Kontext ausgewählt. Die Größe der Schnittmenge kann als Maß für die semantischen Beziehungen angesehen werden. Patwarhan, Banerjee und Pedersen [6] haben herausgefunden, dass das Maß der semantischen Beziehungen durch unterschiedliche Messverfahren bestimmt werden kann.

2.1 Messverfahren

Das Maß der semantischen Beziehung kann bei der Auswahl der richtigen Bedeutung eines Zielwortes im Text helfen. Die Idee dahinter [7] ist die Maximierung des semantischen Verwandtschaftsmaßes. Durch das Maximieren wird die Lösung für WSD optimiert. Für die Bestimmung des Maßes werden folgende Verfahren benutzt:

- Baselines (Grundlinien)
- Path Based Measures (Pfadlängenmaß)
- Information Content Measures (Informationsinhaltsmaß)
- Gloss Based Measures (Überlappungsmaß)

Zum Baseline Verfahren gehören die *Random* Methode, bei der eine Bedeutung zufällig aus den möglichen Bedeutungen gewählt wird und die *Path Length* Methode, in der der kürzeste Pfad zwischen zwei Konzepten in der *is-a* Hierarchie als Maß genommen wird. Auf die Hierarchien in WordNet wird im Abschnitt 3.1 näher eingegangen.

Unter Path Based Measures sind die Methoden von *Wu und Palmer* [10] gemeint. Sie konzentrierten sich auf den Abstand zwischen einem Konzept und dem Wurzelknoten in WordNet Hierarchie. In der Methode von *Leacock und Chodorow* [3] wurde der gleiche Ansatz wie *Path Length* gewählt. Allerdings wurde mit der Taxonomie-Tiefe skaliert.

Die Methoden von *Resnik*, *Lin* und *Jiang - Conrath* gehören zu Information Content Measures. *Resnik* verwendete Informationsgehalt von gemeinsamen Konzepten. *Lin* benutzte den gleichen Ansatz wie *Resnik*, jedoch wurde der Wert mit dem Informationsgehalt der individualen Konzepte skaliert.

Zu Gloss Based Measures gehören die Methoden *Original Lesk*, *Extended Lesk* und *Gloss Vector*. *Extendend Lesk* ist eine Erweiterung des Lesk Algorithmus. Das Lesk Verfahren wurde so angepasst, dass netzwerkbasierte Resourcen wie WordNet verwendet werden können. Hier werden nicht nur die Definitionen der Bedeutungen von zwei Wörtern bei der Berechnung der Überlappung betrachtet. Es werden auch Konzepte, die in einer Beziehung zu den Definitionen stehen, bei der Überlappung berücksichtigt.

Um den *Gloss Vector* zu bilden, muss zuerst eine co-occurrence Matrix aus allen Wörtern des Korpus erstellt werden. Die co-occurrence Matrix enthält die Anzahl, wie oft zwei Wörter gemeinsam in WordNet-Definitionen aufgetreten sind. Hier wurden Stoppwörter und Wörter, die mehr als tausendmal oder weniger als fünfmal auftauchten, nicht für die Erstellung der Matrix berücksichtigt. Jedes Wort wird duch einen Vektor (entweder eine Zeile oder eine Spalte aus der Matrix) repräsentiert. Eine Definition (Gloss) besteht aus mehreren Wörtern, wobei jedes Wort seinen eigenen Vektor hat. Für das Verwandtschaftsmaß müssen zuerst die Glossvektoren bestimmt werden. Der Glossvektor ist der Durchschnittsvektor aus allen Vektoren der Wörter aus dem Gloss. Der Glossvektor stellt die Bedeutung des Konzepts dar. Nach der Erstellung der Glossvektoren für jedes Konzept werden Konzepte paarweise miteinander verglichen, indem der Cosinus des Winkels, der zwischen den Glossvektoren besteht, ermittelt wird.

2.1.1 Formel

Pedersen et. al nutzten in [7] die Gleichung (2.1), um WSD zu beschreiben.

$$\text{argmax}_{i=1}^{m_t} = \sum_{j=1, j\neq t}^{n} \text{max}_{k=1}^{m_j} \text{relatedness}(s_{t,i}, s_{j,k}) \tag{2.1}$$

Wir bezeichnen die Wörter in einem Kontextfenster mit $w_1, w_2, \ldots, w_n$, wobei ein Zielwort w_t mit $1 \leq t \leq n$ genannt wird. Es wird angenommen, dass jedes Wort w_i m_i mögliche Sinne $\{s_{i,1}, s_{i,2}, \ldots, s_{i,m_i}\}$ hat. Um das Verwandschaftsmaß in dem WSD-Algorithmus zu bestimmen, wird die *relatedness*-Funktion verwendet.

$$\text{relatedness} : s_{i,j} \times s_{k,l} \rightarrow R,\ R \in \mathbb{R}$$

Dabei stellen $s_{i,j}$ und $s_{k,l}$ Bedeutungen zweier Wörter aus dem Kontextfenster dar. Die Ausgabe R der *relatedness*-Funktion wird als Indikator für den Verwandschaftsgrad in dem WSD-Algorithmus angenommen.

2.1.2 Expterminte und Ergebnisse

Pedersen et. al benutzten die *Senseval-2 English lexical Sample data*. Diese enthält 4328 Instanzen. Jede Instanz hat einen Satz mit einem Zielwort, das aufgelöst werden muss und ein oder zwei umgebende Sätze, die den Kontext erweitern. Es gibt 73 unterschiedliche Zielwörter, bestehend aus 29 Nomen, 29 Verben und 15 Adjektiven in den verwendeten Daten. Für die 29 Nomen gibt es im Durchschnitt je 8,2 Bedeutungen. Die 29 Verben haben durchschnittlich 12,2 Bedeutungen. Schließlich besitzen die 15 Adjektive im Schnitt

7,1 Bedeutungen. Für jede Instanz wurde das Zielwort in der Mitte positioniert, sofern es möglich war. Im Experiment wurden 6 unterschiedliche Fenstergrößen (2, 3, 5, 11, 21 und 51) getestet.

Es wurden die oben genannten Verfahren für die Messung des Verwandschaftsgrads verwendet. Bei Path based und Information Content Measures wurden nur die Bedeutungen der Nomen ohne Betrachtung der eigentlichen Wortarten berücksichtigt.
Währenddessen wurden bei Hirst - St. Onge und Gloss based Measures alle möglichen Bedeutungen für alle möglichen Wortarten eines Wortes berücksichtigt.

Sie evaluierten ihren Algorithmus, indem sie dessen Ergebnis mit der von Menschen erstellten Auflösung, die als Goldstandard bezeichnet wird, verglichen. Die *Precision*, *Recall*, *Coverage* und *F-Measure* wurden wie folgt berechnet:

$$Precision = \frac{\text{Anzahl der korrekt disambiguierten Wörter}}{\text{Anzahl der disambiguierten Wörter}} \tag{2.2}$$

$$Recall = \frac{\text{Anzahl der korrekt disambiguierten Wörter}}{\text{Anzahl der aufzulösenden Wörter}} \tag{2.3}$$

$$Coverage = \frac{\text{Anzahl der disambiguierten Wörter}}{\text{Anzahl der aufzulösenden Wörter}} \tag{2.4}$$

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.5}$$

Das beste Verfahren ist *Extended Lesk*, weil es die höchsten *F-measure* Werte für alle Wortarten lieferte.

		Fenstergröße					
Wortart	Messverfahren	2	3	5	11	21	51
Nomen	Extended Lesk	0,326	0,377	0,396	0,405	0,412	0,384
Verben	Extended Lesk	0,154	0,198	0,202	0,212	0,201	0,195
Adjektiven	Extended Lesk	0,197	0,245	0,235	0,243	0,232	0,229
Adjektiven	Vektor-based	0,234	0,251	0,240	0,243	0,224	0,212
Alle Wortarten	Extended Lesk	0,231	0,278	0,284	0,291	0,292	0,277

Tabelle 2.1: Die besten F-measures Werte aus [7]

3 Wissensquellen

3.1 WordNet

WordNet ist ein maschinenlesbares Wörterbuch, das an der Princeton Universität entwickelt wurde. Es ist semantisch strukturiert und nicht wie traditionelle Wörterbücher alphabetisch [1]. Das Fundament von WordNet stellen die *Synsets* dar. Ein *Synset* bündelt Synonyme. *Synsets* haben eine eindeutige Identifikationsnummer und eine Definition, die *Gloss* genannt wird. Diese bestehen aus Anwendungsbeispielen und aus einem oder mehreren Sätzen, die die Bedeutung beschreiben. Des Weiteren sind *Synsets* miteinander durch Relationen verknüpft. Ein *Synset* enthält genau nur eine Bedeutung für ein Wort. Somit würde ein mehrdeutiges Wort in unterschiedlichen *Synsets* auftreten.

In WordNet sind verschiedene Wortarten wie Adjektive, Adverbien, Nomen und Verben vertreten. WordNet ist hierarchisch strukturiert. Die Verknüpfungen der *Synsets* geben Information darüber, welche Relation die *Synsets* zueinander haben. Die Relationen unterscheiden sich nach den Wortarten.

Die Relationen *Hyponym*, *Hyperonym*, *Meronym* und *Holonym* sind für Nomen vorhanden. Ein Wort W_1 ist ein *Hyponym* vom Wort W_2, wenn W_1 eine Spezialisierung von W_2 ist. Umgekehrt ist W_2 ein *Hyperonym* von W_1, wenn W_2 eine Generalisierung von W_1 ist. Ist W_1 ein Teil von W_2, so ist W_1 *Meronym* von W_2. Andersherum ist W_2 ein *Holonym* von W_1.

Für Verben existieren die Relationen *Hyperonym*, *Cause* und *Entailment*. Zum Beispiel impliziert das Verb „schnarchen“ das Verb „schlafen“. Diese Relation heißt *Entailment*. Bei Verben gibt es auch die Relation Spezialisierung und Generalisierung.

Für Adjektive sind die Relationen *Similarity*, *Pertainymy*, *Attribute* und *See-also*.

- *Similarity*: ein Adjektiv A_1 ist ähnlich zu einem anderen Adjektiv A_2, z. B. schön ist ähnlich zu hübsch.
- *Attribute*: ein Nomen N_1 ist ein Attribut für eine Eigenschaft A_1, die einen Wert beschreibt, z. B. heiß ist ein Wert für Temperatur.
- *See-also*: ist eine Relation zwischen verwandten Eigenschaften, z. B. schön ist verwandt mit der Eigenschaft attraktiv durch die Relation *See-also*.
- *Pertainymy*: gehört ein Adjektiv A_1 zu einem Nomen N_1, so ist A_1 *pertainym* zu N_1, z. B. medizinisch ist pertainym zu Medizin

Die erwähnten Relationen und deren Bedeutungen wurden von der Quelle [5] entnommen. Mehr über die Relationen kann man auf der Projektwebseite[1] oder in der Arbeit [9] nachlesen.

[1] https://wordnet.princeton.edu

3.2 Dataset

Um der Algorithmus zu evaluieren, wurden zwei Korpora in englischer Sprache verwendet.

Das erste Korpus ist aus dem SemEval 2010 Task-17 „All-words Word Sense Disambiguation on a Specific Domain (WSD-domain)“. Es besteht aus drei Texten aus dem Bereich Umwelt mit insgesamt 3261 Wörter, davon sind 1398 aufzulösenden Wörter (Target-Words). Diese Texte stammen aus Artikeln des WWF [2] und ECNC [3]. Das Korpus ist in einer XML-Datei mit einer Struktur wie in Abbildung 3.1 gespeichert.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE corpus SYSTEM  "../all-words.dtd">
3  <corpus lang="en">
4    <text id="en1">
5      <s>
6        a.
7        Increasing
8        the
9        <head id="en1.s002.t23">area</head>
10       of
11       National
12       Ecological
13       <head id="en1.s002.t27">Network</head>
14       further
15       .
16     </s>
17   </text>
18 </corpus>
```

Abbildung 3.1: Aufbau des Korpus von Task-17

In dem Korpus gibt es drei Texte. Jeder Text besteht aus mehreren Sätzen, die mit dem **s**-Tag eingeschlossen sind. Die aufzulösenden Wörter sind mit **head**-Tags markiert. In den Attributen des **head**-Tags ist die eindeutige ID des Wortes zu finden. Wörter, die es nicht zu disambiguieren gilt, sind nicht mit Tags umschlossen.

Das zweite Korpus besteht nur aus einem Text mit insgesamt 677 aufzulösenden Wörtern, der aus dem Informatikbereich stammt. Es ist aus dem SemEval 2007 Task-7. Das Korpus ist in einer XML-Datei mit einer Struktur wie in Abbildung 3.2 gespeichert. In dem Korpus gibt es nur einen Text. Der Text besteht aus mehreren Sätzen, die mit dem **sentence**-Tag eingeschlossen sind. Die aufzulösenden Wörter sind mit **instance**-Tags markiert. In den Attributen des **instance**-Tags sind die eindeutige ID, die Grundform und die Wortart des Wortes enthalten. Wörter, die es nicht zu disambiguieren gilt, sind nicht mit Tags

[2] Der WWF (World Wide Fund for Nature, bis 1986 World Wildlife Fund) ist eine Schweizer Stiftung mit Sitz in Gland, Kanton Waadt. Sie wurde 1961 gegründet und ist eine der größten internationalen Natur- und Umweltschutzorganisationen. [http://www.wwf.de]

[3] European Centre for Nature Conservation (Kurzform ECNC) ist eine europäische Naturschutzorganisation mit Sitz in den Niederlanden. Sie wurde 1993 gegründet. [http://www.ecnc.org]

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<corpus lang="en">
  <text id="d004">
    <sentence id="d004.s001">
      <instance id="d004.s001.t001" lemma="Computer" pos="n">
        Computer
      </instance>
      <instance id="d004.s001.t002" lemma="programming" pos="n">
        programming
      </instance>
      is
      intresting
    </sentence>
  </text>
</corpus>
```

Abbildung 3.2: Aufbau des Korpus aus Task 7

umschlossen.
Alle aufzulösenden Wörter wurden gemäß dem Kontext des Korpus von einem Menschen disambiguiert und das Ergebnis in einer Ergebnisdatei, Goldstandard genannt, gespeichert. Der Goldstandard dient als eine Referenz zum Vergleichen mit dem Ergebnis des Algorithmus.

Die Ergebnisdatei des ersten Korpus aus Task-17 hat in jeder Zeile zwei wesentliche Einträge. Der erste Eintrag enthält die eindeutige Wort-ID und in dem zweiten Eintrag steht genau eine WordNet-ID der passenden Bedeutung.

In der Ergebnisdatei des zweiten Korpus aus Task-7 (D004) stehen auch die IDs der aufzulösenden Wörter und die WordNet-ID der passenden Bedeutung, aber es könnten mehrere WordNet-IDs angegeben werden.

4 Simulated Annealing

Simulated Annealing wird bei der Lösung von kombinatorischen sehr komplexen Minimierungsproblemen angewendet. Sehr komplexe Aufgaben der Kombinatorik, wie das Problem des Handlungsreisenden, wurden mit Simulated Annealing gelöst. Die zu lösende Aufgabe des Handlungsreisenden ist die Suche nach einem Weg, bei dem eine Anzahl von Städten mit minimalen Kosten (Geld, Zeit, Strecke) besucht werden soll. Der Algorithmus ist eine Anlehnung an einen physikalischen Prozess, bei dem Stahl langsam abgekühlt und wieder erhitzt wird, damit er eine gleichmäßige Struktur erhält, die ihm seine Härte verleiht. Dabei erreicht der Stahl seinen minimalen Energiestatus. Im Englischen heißt dieser Prozess Annealing.

Simulated Annealing wird in NLP bei Word Sense Disambiguation eingesetzt, weil der Algorithmus besondere Eigenschaften besitzt, wie z.B. das gleichzeitige Disambiguieren mehrere Wörter, während die meisten WSD Algorithmen Wort für Wort nacheinander disambiguieren. Des Weiteren sucht Simulated Annealing in der lokalen Umgebung ein Optimum, jedoch wird die lokale Suche in einer globalen Suche durch eine Zufallsauswahl und eine monoton fallende Wahrscheinlichkeitsfunktion erweitert.

4.1 Simulated Annealing für generelle Domain

Gegeben ist ein Satz s, der aus mehreren Wörtern $\{w_1, w_2, \ldots, w_n\}$ besteht. Jedes Wort w_i hat mehrere Bedeutungen $\{m_1, m_2, \ldots, m_{n_i}\}$. , wobei n_i die Anzahl der Bedeutungen für das i-te Wort in dem maschinenlesbaren Wörterbuch ist. Eine Konfiguration $C = \{m_{1,j}, m_{2,k}, \ldots, m_{n,l}\}$ ist die Auswahl einer Bedeutung für jedes Wort w_i aus dem Satz.

Am Anfang wird zufällig eine Konfiguration C für den Satz ausgewählt. Als Beispiel in der Arbeit [2] von *Jim Cowie, Joe Guthrie and Louise Guthrie* wurde die initiale Konfiguration C so ausgewählt, dass jedes Wort seine erste Bedeutung zugewiesen wurde und die Temperatur T einen initialen Wert von 1 hat.

Der Parameter T wird langsam bei jeder Iteration durch $T = \alpha * T$ abgekühlt. Die Abkühlung geschieht so lange, bis das System sein Minimum erreicht hat. Des Weiteren wird eine Energiefunktion E_c benötigt, die die Energie zur gegebenen Konfiguration C berechnet, indem man die Redundanz R für die Konfiguration bestimmt.

Die Wörter der Bedeutungen werden in einer Liste mit den N aufzulösenden Wörtern eingefügt. Die Anzahl n_i der Erscheinung jedes Terms t_i in der Liste wird summiert. Die Redundanz berechnet sich aus

$$R = \sum_{0}^{i} (n_i - 1).$$

Somit lässt sich die Energie durch die Formel

$$E = \frac{1}{1 + R}$$

berechnen. Durch Erhöhung der Redundanz wird die Energie minimiert.

Danach wird eine neue Konfiguration C' zufällig ausgewählt und deren Energiewert berechnet. Ist die neue $E_{c'}$ kleiner als die vorherige E_c, so ersetzt die neue Konfiguration C' die alte C. Sonst wird durch eine Wahrscheinlichkeitsfunktion $P(\Delta E, T) = exp(-\frac{\Delta E}{T})$ entschieden, ob die „schlechte“ neue Konfiguration C' die alte C ersetzt oder nicht. Diese Entscheidung wird durch einen Zufallszahlengenerator getroffen. Falls die generierte Zahl kleiner als P ist, ersetzt die neue Konfiguration C' die alte C.

4.1.1 Vorbereitung der Eingabedaten

Als erstes müssen die Eingabedaten für die Implementierung des eigentlichen Algorithmus vorbereitet werden, da das Korpus in einer XML-Datei vorgegeben ist. Um auf den Inhalt des Korpus, sprich die Texte, die Sätze und die einzelnen Wörter zugreifen zu können, muss erst einmal der Inhalt der XML-Datei für die darüber liegenden Schichten der Anwendung in einer entsprechend passenden Form zur Verfügung gestellt werden. Die Gewinnung dieser Informationen geschieht in aufeinander folgenden Schritten. Der erste Schritt ist das Parsen der Datei in einer Objektstruktur, wie in Abbildung 4.1 zusehen ist.

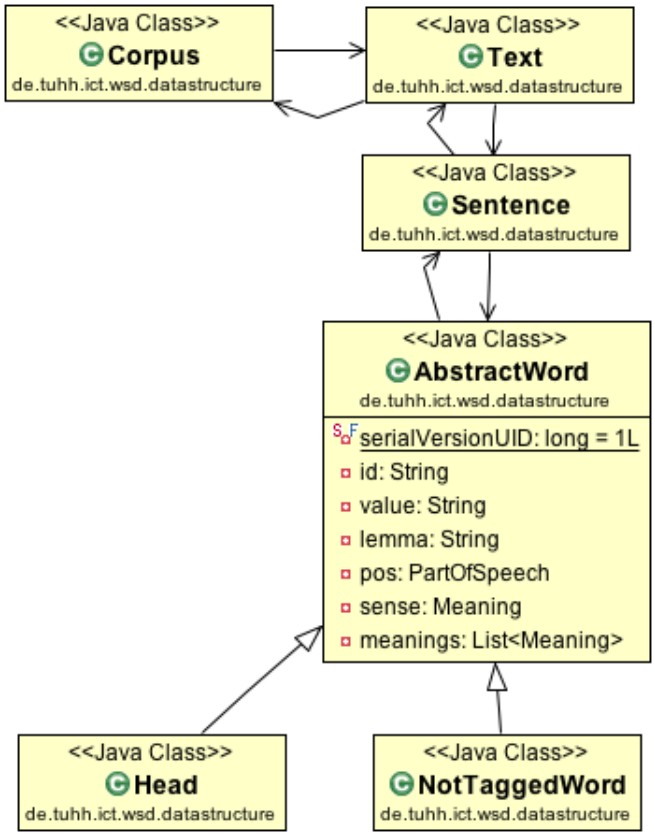


Abbildung 4.1: Vereinfachte UML-Darstellung des Korpus

Die AbstractWord-Klasse in der Implementierung hat Lemma, das unveränderte Wort und die Wortart als Felder. Während des Parsens wird bereits jedes Wort lemmatisiert. Dabei werden die Wörter in ihre Grundform gebracht. Zum Beispiel wird das Wort „lessons“ durch das Lemmatisieren zu „lesson“ umgewandelt. Dieser Schritt ist notwendig, da die Wörter in WordNet nur in der Grundform gefunden werden können.

Nachdem ein Satz in einer Instanz Sentence erstellt wurde, werden die Wortarten für den gesamten Satz, bestehend aus den nicht-lemmatisierten Wörtern, bestimmt. Hierfür wurden die Wortarten durch Stanford CoreNLP und WordNet bestimmt. Die Klasse Sentence hat vier wesentlichen Felder, eine Liste der aufzulösenden Wörter, ein Liste der nicht-aufzulösenden Wörter, eine Liste mit allen Wörtern und eine Referenz auf dem Text-Objekt, zudem der Satz gehört. Die verwendeten Listen erhalten die Reihenfolge der eingefügten Wörter. Nachfolgend werden alle Stoppwörter aus dem Satz entfernt, wie z.B.„and“, „the“, „of“, „this“, etc. Schließlich werden alle möglichen Bedeutungen für jedes aufzulösende Wort aus WordNet geholt und den jeweiligen Wörtern zugewiesen. Somit hat jedes Wortobjekt eine Liste mit seinen möglichen Bedeutungen.

4.1.2 Algorithmus 1: Standard Implementierung

Um die Ergebnisse vom Abschnitt 4.2 später besser analysieren zu können, beginnen wir mit dem Vorstellen der Implementierung von Simulated Annealing für generelle Domain.

In der Standardimplementierung wurde die Initialkonfiguration C aus der ersten Bedeutung der jeweiligen Wörter gebildet. Der Initialwert für die Temperatur ist stets eins. Die Konfiguration C' besteht aus zufällig ausgewählten Bedeutungen, d.h., für jedes Wort des Satzes wird eine zufällige Bedeutung aus deren möglichen Bedeutungen ausgewählt. Bei der Energieberechnung, wie in Pseudocode 1, wurden nur die Lemmata der Wörter, die Lemmata der Bedeutungen sowie die Lemmata der verwandten Synsets, die durch bestimmte Relationen (Siehe Abschnitt 3.1) mit den Bedeutungen verbunden sind, berücksichtigt. Es wurde eine Liste, die aus allen Lemmata der Wörter des Satzes und allen Wörtern der Bedeutungen aus der Konfiguration besteht, für die Berechnung der Redundanz erstellt.

Algorithm 1: Simulated Annealing für generelle Domain

Input : $sentence = [w_1, w_2, \cdots, w_n]$
Output : $bestMeanings = [m_1, m_2, \cdots, m_n]$

begin
  $T \leftarrow 1$
  $\alpha \leftarrow 0.9$
  $maxNrIter \leftarrow MAX\_ITER$
  $maxNrSameIter \leftarrow MAX\_SAMEITER$
  $bestMeanings \leftarrow [\,]$
  $bestEnergy \leftarrow 1$
  `/* Begin initial/first Configuration */`
  for $\forall w_i \in sentence$ **do**
    $Ms \leftarrow allMeaningsOfCurrentWord(w_i)$
    if $notEmpty(Ms)$ **then**
      $bestMeanings[i] \leftarrow Ms[0]$
  `/* End initial/first Meanings */`
  $bestEnergy \leftarrow energy(sentence, bestMeanings)$
  while $maxNrIter > 0$ **and** $maxNrSameIter > 0$ **do**
    `/* Begin next configuration */`
    $currentMeanings \leftarrow [\,]$
    $currentEnergy \leftarrow 1$
    for $\forall w_i \in sentence$ **do**
      $Ms \leftarrow allMeaningsOfCurrentWord(w_i)$
      if $notEmpty(Ms)$ **then**
        $meaningNr \leftarrow randomNr(0, length(Ms))$
        $currentMeanings[i] \leftarrow Ms[meaningNr]$
    `/* End next configuration */`
    $currentEnergy \leftarrow energy(sentence, currentMeanings)$
    if $currentEnergy = bestEnergy$ **then**
      $maxNrSameIter \leftarrow (maxNrSameIter - 1)$
    if $currentEnergy < bestEnergy$ **then**
      $bestEnergy \leftarrow currentEnergy$
      $bestMeanings \leftarrow currentMeanings$
      $maxNrSameIter \leftarrow MAX\_SAMEITER$
    else
      $P \leftarrow exp(\,-(bestEnergy - currentEnergy)/T\,)$
      $randNr \leftarrow randNumber()$
      if $randNr < P$ **then**
        $bestEnergy \leftarrow currentEnergy$
        $bestMeanings \leftarrow currentMeanings$
        $maxNrSameIter \leftarrow MAX\_SAMEITER$
    $T \leftarrow \alpha * T$
    $maxNrIter \leftarrow (maxNrIter - 1)$

Pseudocode 1: Energieberechnung für generelle Domain WSD

```
/* Energieberechnung                                                    */
listOfWord ← [];
j ← 0;
/* Füge alle Lemmas der Wörter aus dem Satz ein                         */
for ∀w_i ∈ sentence do
    listOfWord[j++] ← lemma(w_i);
/* Füge alle Lemmata aus den Glossen der einzelnen Bedeutungen und alle
   Lemmata der Glossen von den verwandten Bedeutungen ein               */
for ∀m_i ∈ meanings do
    for ∀lemma ∈ gloss(m_i) do
        listOfWord[j++] ← lemma;
    for ∀m_k ∈ relatedMeanings(m_i) do
        for ∀lemma ∈ gloss(m_k) do
            listOfWord[j++] ← lemma;
/* Berechne die totale Redundanz R                                      */
R ← 0;
for ∀term_i ∈ listOfWord do
    sum ← 0;
    for ∀str_l ∈ listOfWord do
        if term_i = str_l then
            sum ← (sum + 1);
    R ← R + (sum − 1);
/* Energie                                                              */
energy ← 1/(1 + R)
```

4.2 Simulated Annealing für spezifische Domänen

Die meisten Aufgaben von Word Sense Disambiguation waren auf generelle Domain, d. h. nicht spezifische Domain, fokussiert. Die Adaptation von spezifischen Domänen ist eine aktuelle Problemstellung in NLP insbesondere im Bereich von Word Sense Disambiguation.

Eine Herausforderung bei der Auflösung von sprachlichen Mehrdeutigkeiten für spezifische Domänen ist die Erstellung der Domain. Eine weitere Schwierigkeit ist die Berücksichtigung der Domain während des Disambiguierens.

In diesem Abschnitt beschäftigen wir uns mit zwei Ansätzen der Word Sense Disambiguation für spezifische Domänen mit Hilfe von Simulated Annealing.

4.2.1 Erstellung der Domain

In dieser Arbeit wurde die Domain als eine Tabelle mit zwei Spalten, in der die Schlagwörter und ihre Auftrittshäufigkeiten eingetragen wurden, implementiert. Jede Bedeutung wird in WordNet durch ein Synset dargestellt und jedes Synset hat eine Menge von Schlagwörtern. In WordNet sind die Synsets miteinander durch sogenannte Relationen bzw. Pointers verbunden. Die domainnahen Relationen wie Domain, Topic, Topic-Member, Usage, Usage-Member, Region und Region-Member in WordNet wurden für die Generierung der Domain für jede mögliche Bedeutung berücksichtigt.

Die Auftrittshäufigkeit jedes Schlagwortes berechnet sich aus der Anzahl ihrer Wiederholungen in den Synsets, mit denen die Bedeutung durch bestimmte Relationen verbunden ist.

Die Domain eines Textes berechnet sich aus der Summe der Domänen aller aufzulösenden Wörter im Text. Nachdem die Textdomain bestimmt wurde, werden nur die drei am häufigsten wiederholten Schlagwörter ausgewählt, die somit die Domain des Textes bildet. Die Domänen der einzelnen Bedeutungen wurden während des Holens der möglichen Bedeutungen für jedes Wort generiert. Bei der Auswahl der Bedeutungen und bei der Berechnung der Energie wurde nur die Domain eines Textes berücksichtigt. Nun richten wir unser Augenmerk in den folgenden beiden Abschnitten genauer auf WSD für spezifische Domänen.

4.2.2 Algorithmus 2

Der Algorithmus aus dem ersten Fall wurde nur an drei verschiedenen Stellen so angepasst, dass die Domain des Textes großen Einfluss auf die Auswahl der besten Bedeutung hat.

Bestimmen der Initialkonfiguration
Die Initialkonfiguration besteht jetzt nicht aus den ersten Bedeutungen der aufzulösenden Wörtern, sondern aus den Bedeutungen, die die größte Überlappung mit der Domain des Textes haben, wie in Pseudocode 2 beschrieben ist.

Pseudocode 2: Initialkonfiguration (Simulated Annealing WSD für spezifische Domain)

Input : $sentence$
Output : $initialConfiguration$
`/* Initial Konfiguration */`

```
1  for ∀w_i ∈ sentence do
2      Ms ← allMeaningsOfCurrentWord(w_i);
3      bestFitMeaning ← Ms[0];
4      bestOverlap ← score(textDomain, sentence, bestFitMeaning);
5      for ∀m_i ∈ Ms do
6          currentOverlap ← overlapMeasure(textDomain, sentence, m_i);
7          if currentOverlap > bestOverlap then
8              bestOverlap ← currentOverlap;
9              bestFitMeaning ← m_i;
10     bestOverlaps[i] ← bestOverlap;
11     initialConfiguration[i] ← bestFitMeaning;
```

Berechnung der nächsten Konfiguration
Beim Bestimmen der nächsten Konfiguration hat die Domain eine wichtige Rolle. Der Überlappungswert der vorherigen Bedeutung wurde in einer Liste gespeichert, damit der Überlappungswert beim Bestimmen der nächsten Konfiguration verwendet werden kann. Für jedes Wort aus dem Satz sucht man zufällig eine Bedeutung $randomMeaning$ von seiner möglichen n Bedeutungen. Danach wird die Überlappung von $randomMeaning$ mit der TextDomain und dem Satz berechnet. Ist die Überlappung größer als die Überlappung der vorherigen Bedeutung, so wird $randomMeaning$ als die passende Bedeutung für das Wort in der aktuelle Konfiguration ausgewählt. Diese Suche nach der besten überlappenden Bedeutung mit der Textdomain und dem Satz wird n mal wiederholt. Wird keine Bedeutung mit höherer Überlappung gefunden, so wird die letzte zufällig ausgewählte Bedeutung für das Wort in der aktuellen Konfiguration genommen.

Pseudocode 3: Bestimmen der nächsten Konfiguration für WSD für spezifische Domain

Input : $sentence, oldOverlap$
Output : $currentMeanings$

```
/* Bestimmung der nächsten Konfiguration                                    */
1  for ∀w_i ∈ sentence do
2      Ms ← allMeaningsOfCurrentWord(w_i);
3      bestFitMeaning ← Ms[0];
4      bestOverlap ← oldOverlap[i];
5      randomMeaning ← null;
6      currentOverlap ← 0;
7      found ← false;
8      for j ∈ {1, ..., |Ms|} do
9          meaningNr ← randomNr(0, length(Ms));
10         randomMeaning ← Ms[meaningNr];
11         currentOverlap ← score(textDomain, sentence, randomMeaning);
12         if bestOverlap < currentOverlap then
13             currentMeanings[i] ← randomMeaning;
14             bestOverlap ← currentOverlap;
15             found ← true;
16             break;
17     if not found then
18         currentMeanings[i] ← randomMeaning;
19         bestOverlap ← currentOverlap;
```

Energieberechnung

Hier hat die Domain auch eine Bedeutung bei der Energieberechnung. Im Wesentlichen ist die Berechnung der Energie sehr ähnlich zu der bei generellen Domain verwendeten Berechnung. Kommt jedoch ein Term aus der Liste der Wörter in der Textdomain vor, so wird diese mit einer sehr großen Zahl bewertet. Im Pseudocode 4 in den Zeilen 17 und 18 wird die Berücksichtigung der Textdomain dargestellt.

Pseudocode 4: Energieberechnung für WSD für spezifische Domain

```
Input  : sentence, Configuration
Output : energy
/* Energieberechnung                                                    */
listOfWord ← [];
j ← 0;
/* Füge alle Lemmas der Wörter aus dem Satz                             */
for ∀w_i ∈ sentence do
    listOfWord[j++] ← lemma(w_i);
/* Füge alle Lemmata aus den Glossen der einzelnen Bedeutungen und alle
   Lemmata der Glossen von den verwandten Bedeutungen ein               */
for ∀m_i ∈ Configuration do
    for ∀lemma ∈ gloss(m_i) do
        listOfWord[j++] ← lemma;
    for ∀m_k ∈ relatedMeanings(m_i) do
        for ∀lemma ∈ gloss(m_k) do
            listOfWord[j++] ← lemma;
/* Berechne die totale Redundanz R                                      */
R ← 0;
for ∀term_i ∈ listOfWord do
    sum ← 0;
    for ∀str_l ∈ listOfWord do
        if term_i = str_l then
            sum ← (sum + 1);
    /* Berücksichtigung der Textdomain beim berechnen der Redundanz     */
    if isInDomain(term_i) then
        sum ← (sum + 10);
    R ← R + (sum − 1);
/* Energie                                                              */
energy ← 1/(1 + R)
```

4.2.3 Algorithmus 3

Die nächste Alternativimplementierung hat einen wesentlichen Unterschied zu den vorherigen Implementierungen, sowohl für generelle Domain als auch für spezifische Domain. In diesem Fall werden zu den aufzulösenden Wörtern $\{w_1, \cdots, w_n\}$ des Satzes zusätzlich die drei Schlagwörter der Textdomain als aufzulösende Wörter $\{d_1, d_2, d_3\}$ zu der Liste der aufzulösenden Wörter hinzugefügt. Zu jedem Satz werden während der Disambiguierung die drei Schlagwörter der Textdomain hinzugefügt und nach der Disambiguierung entfernt. Zwei weitere Unterschiede zu der vorherigen WSD-Implementierung für spezifische Domain bestehen zum einen bei der Erstellung der Initialkonfiguration, zum anderen bei der Bestimmung der nächsten Konfiguration.

Bei der Initialkonfiguration werden, wie bei WSD für generelle Domain, nur die ersten Bedeutungen der Wörtern ausgewählt. Bei der Bestimmung der nächsten Konfiguration wird nur ein Wort aus der Liste der aufzulösenden Wörter zufällig ausgewählt. Zu diesem zufällig ausgewählten Wort wird eine Bedeutung m_{random} ebenfalls zufällig aus seinen möglichen n Bedeutungen ausgewählt. Anhand des Überlappungswertes von m_{random} mit der Textdomain und dem Satz wird entschieden, ob m_{random} die vorherigen Bedeutung ersetzt. Ersetzt m_{random} die vorherige Bedeutung nicht, so wird erneut eine Bedeutung m_{random} zufällig ausgewählt und entsprechend verglichen. Wenn keine passende Bedeutung mit höherem Überlappungswert als dem vorherigen gefunden wurde, wird dieser Prozess n mal wiederholt. Wird nach n Iterationen keine passende Bedeutung gefunden, wird die zuletzt zufällig ausgewählte Bedeutung genommen.

Nachdem alle Iterationen ausgeführt wurden, wird die zuletzt ausgewählte beste Konfiguration als die passende Bedeutung für den Satz eingesetzt.

5 Ergebnisse und Analyse

Die Algorithmen wurden in der Programmiersprache Java implementiert. In Java Standardbibliothek ist eine Implementierung der „Document Object Model (DOM) Level 3 Core Specification“ vorhanden. Diese Implementierung wurde zum Parsen der Korpora verwendet, da die Korpora in XML-Format bereitgestellt wurden. Um auf WordNet zugreifen zu können, wurde die „ Java Wordnet Interface“ (JWI) Bibliothek von MIT[1] genutzt. Die aufzulösenden Wörter vom Korpus aus Task 17 sind nicht mit Wortarten und Grundformen versehen. Daher musste eine weitere Bibliothek zum Erkennen der Wortarten und zum Bestimmen der Grundformen verwendet werden. Diese Bibliothek heißt Stanford CoreNLP und stammt von der Stanford Natural Language Processing Group.

5.1 Experimente und Ergebnisse

Für die Evaluation der Algorithmen wurden das Korpus aus SemEval-2010 Task 17 und das Korpus aus SemEval 2007 Task 7 verwendet. Das Korpus aus Task 17 hat 1398 aufzulösende Wörter, davon sind 1032 Nomen und 366 Verben. Das Korpus aus Task 7 hat 677 aufzulösende Wörter, von denen 377 Nomen, 145 Verben, 104 Adjektiven und 51 Adverbien sind. Damit wir die Ergebnisse von unserem System mit den Teilnehmern aus Task 17 vergleichen können, werden die Maße *Precision*, *Recall*, *Coverage* und *F-Measure* aus den Gleichungen 2.2, 2.3,2.4 und 2.5 berechnet.

Anzahl der aufzulösenden Wörter und Anzahl der aufgelösten Wörter sind in beiden Korpora gleich, daher ist der *Coverage*-Wert bei beiden Korpora gleich 100%. Somit sind die Werte für *Precision*, *Recall* und *F-Measure* gleich.

Nun wird die Performance des Systems gezeigt, indem die Ergebnisse mit den Grundlinien und untereinander verglichen werden. Die Grundlinien sind die Ergebnisse der Evaluation aus zwei unterschiedlichen Ansätzen. Die Ansätze sind folgende:

- man wählt für jedes aufzulösende Wort nur die erste Bedeutung aus WordNet
- man wählt für jedes aufzulösende Wort zufällig eine Bedeutung aus WordNet

Die Resultate der Grundlinien in Tabelle 5.1 zeigen keinen eindeutigen Trend, denn der Ansatz mit der ersten Bedeutung erzielt bessere Ergebnisse als alle drei implementierten Algorithmen. Dabei hat der Ansatz mit den zufällig ausgewählten Bedeutungen die schlechteste Präzision erreicht.

	Korpus aus Task 17	Korpus aus Task 7
Erste Bedeutung	0,5	0,75
Zufällige Bedeutung	0,23	0,61

Tabelle 5.1: Precision der Grundlinien

[1] Massachusetts Institute of Technology

	Korpus aus Task 17	Korpus aus Task 7
Algorithmus 1	0,24	0,68
Algorithmus 2	0,25	0,69
Algorithmus 3	0,40	0,71

Tabelle 5.2: Precision der Algorithmen für die zwei Korpora

Die Ergebnisse des ersten Algorithmus aus Tabelle 5.2 sind, wie erwartet, kleiner als die übrigen, denn im ersten Algorithmus wurde die Domain während der Disambiguierung nicht berücksichtigt. Die Korpora gehörten jedoch jeweils zu einer Domain, was ein Grund für das niedrige Ergebnis sein kann.

Da die Aufgabe die Entwicklung eines WSD-Algorithmus mit Simulated Annealing für spezifische Domänen ist, wurden die Algorithmen 2 und 3 erstellt. Der zweite Algorithmus sollte eigentlich höhere *Precision*-Werte liefern als der erste, da die Domain beim Disambiguieren betrachtet wurde und einen Einfluss auf die Ergebnisse haben sollte.

Wie in Tabelle 5.2 zu sehen ist, sind die Ergebnisse jedoch nicht gravierend besser als die Ergebnisse des ersten Algorithmus. Somit waren Verbesserungen am zweiten Algorithmus erforderlich. Aus den Veränderungen des zweiten Algorithmus entstand der dritte Algorithmus (siehe Abschnitt 4.2.3). Man bemerkt für das Korpus aus Task 17 einen Anstieg von 60% der Genauigkeit bei der Auflösung zwischen dem zweiten und dem dritten Algorithmus. Diese Verbesserungen erhöhten die Genauigkeit beim Disambiguieren.

Eine bemerkenswerte Tatsache ist, dass die Ergebnisse des Korpus aus Task 7 viel höher ausfallen als die Ergebnisse für das Korpus aus Task 17. Zum Beispiel ist das Ergebnis des ersten Algorithmus für das Korpus aus Task 7 um 183,33% höher ist, als für das Korpus aus Task 17. Der Grund hierfür ist, dass die aufzulösenden Wörter aus Task 7 mit Grundform und Wortart versehen sind. Für das Korpus aus Task 17 sind die aufzulösenden Wörter jedoch nicht mit Grundform und Wortart versehen. Dies hat zur Folge, dass die Grundformen und Wortarten durch Stanford CoreNLP und WordNet bestimmt werden müssen. Die Auflösung der Wortarten und der Grundformen sind nicht immer korrekt. Somit ist die Auswahl der passenden Bedeutung beim Disambiguieren öfter falsch. Der zweite Grund liegt in der Tatsache, dass im Korpus aus Task 17 nur Nomen und Verben aufzulösen sind. Aber im Korpus aus Task 7 ist das Spektrum der Wortarten von den aufzulösenden Wörter größer, da er um die Wortarten Adjektive, Adverbien erweitert wurde.

Die Domain beim dritten Algorithmus spielt eine dominantere Rolle als beim zweiten Algorithmus. Die Anzahl der Iterationen beim dritten Algorithmus hat auch einen entscheidenden Einfluss auf die Genauigkeit. Je kleiner die Anzahl der Iterationen und der Abkühlungsfaktor sind, desto höher ist die Genauigkeit des Disambiguierens. Der zweite Algorithmus liefert bessere Ergebnisse, wenn die Anzahl der Iterationen hoch und der Abkühlungsfaktor niedrig sind.

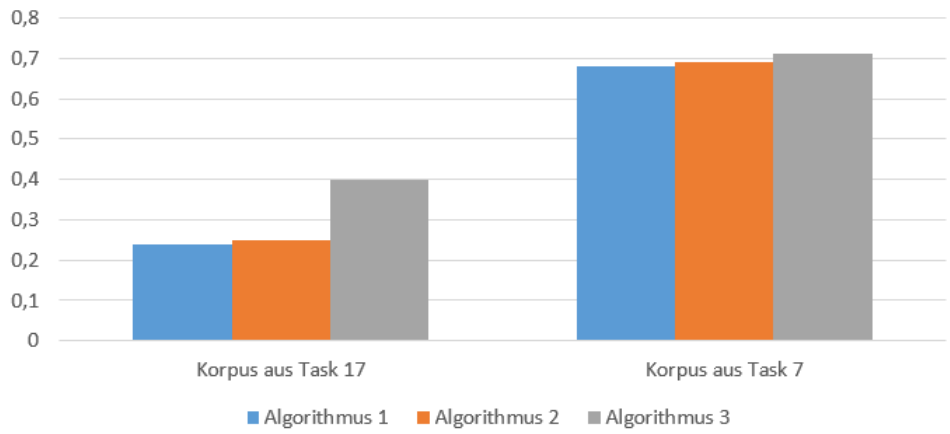


Abbildung 5.1: Precision der Algorithmen je Korpus

5.2 Detaillierte Analyse

5.2.1 Pro Text

Alle Algorithmen konnten den dritten Text am besten disambiguieren, was in Tabelle 5.3 und in Abbildung 5.2 gut zuerkennen ist. Für die Algorithmen zwei und drei wurde stets das gleiche Verfahren genutzt, um die Domain für die einzelne Texte zu generieren. Die Mehrheit der aufzulösenden Wörter aus dem dritten Text sind sehr nah an Umweltkontext. Somit konnte eine sehr viel präzisere Domain für den dritten Text erstellt werden als für die übrigen Texte. Die Domain für den dritten Text enthielt die Schlagwörter *science*, *biology* und *biological* mit den Auftrittshäufigkeiten 49, 41 und 41. Dies war der entscheidende Faktor für das bessere Resultat.

	Text 1	Text 2	Text 3
Algorithmus 1	0,24	0,20	0,27
Algorithmus 2	0,23	0,21	0,31
Algorithmus 3	0,38	0,38	0,45

Tabelle 5.3: Precision der Algorithmen für die jeweiligen Texte des Korpus aus Task 17

Der erste Text konnte besser disambiguiert werden als der zweite Text, da die Domain des ersten Textes nicht so gestreut, wie es beim zweiten Text der Fall war. Denn tritt z. B. das Schlagwort *science* der Domain des ersten Textes dort 35 Mal auf, während es auch zur Domain des zweiten Textes gehört, dort aber nur 14 Mal auftritt. Das veranschaulicht, dass die Domain eine große Rolle beim Disambiguieren spielt.

Das verwendete Korpus aus Task 7 besteht aus einem Text, somit kann eine detaillierte Analyse aus dem Abschnitt 5.1 entnommen werden.

5.2.2 Pro Wortart

In Abbildung 5.3 sind zwei Merkmale festzuhalten. Die Nomen werden mit höherer Genauigkeit disambiguiert als die Verben. Der Grund hierfür ist, dass in WordNet 3.0 117097 Nomen und 11488 Verben enthalten sind. Es ist ersichtlich, dass Nomen besser disambiguiert werden

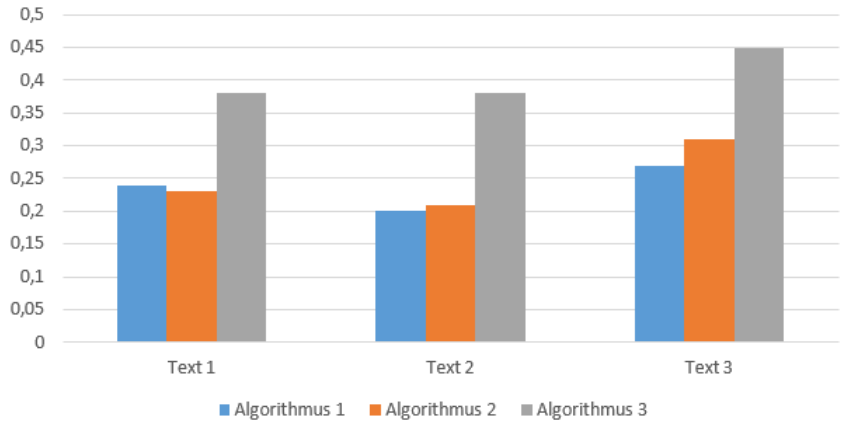


Abbildung 5.2: Precision der Algorithmen je Text aus dem Korpus aus Task 17

können als Verben. Das zweite Merkmal ist, dass der dritte Algorithmus die Mehrdeutigkeit besser auflöst als die übrigen zwei Algorithmen.

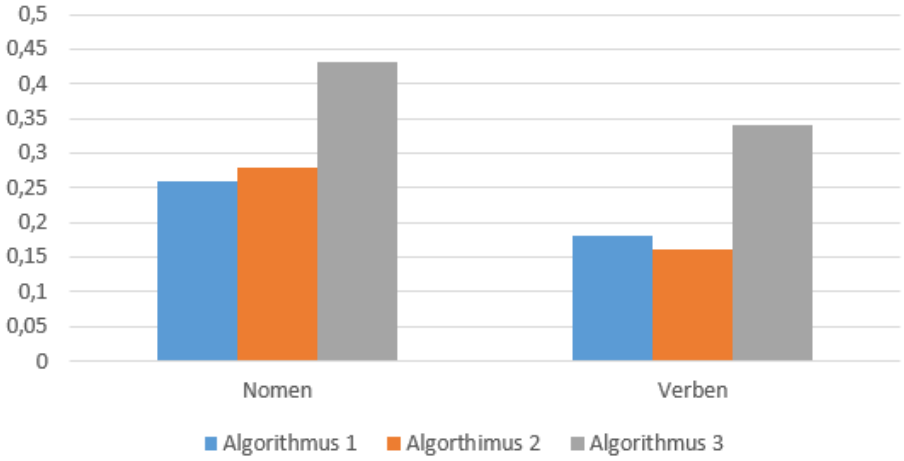


Abbildung 5.3: Precision der Algorithmen je Wortart aus dem Korpus aus Task 17

Im Korpus aus Task 7 werden die Adjektive und Adverbien etwas genauer aufgelöst als die Nomen und die Verben. Aber man sieht keinen großen Unterschied zwischen den Ergebnissen der Wortarten untereinander, wie es in Abbildung 5.4 zusehen ist. Hier zeigt der dritte Algorithmus seine bessere Performance.

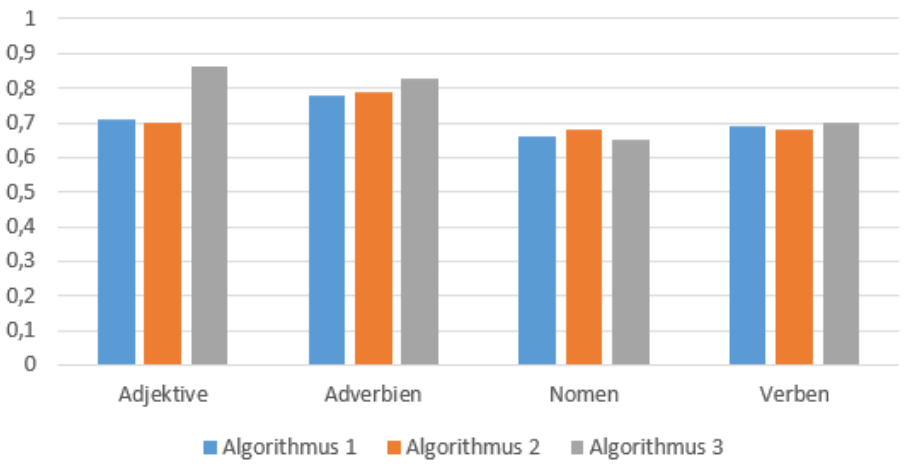


Abbildung 5.4: Precision der Algorithmen je Wortart aus dem Korpus aus Task 7

5.3 Vergleich mit Algorithmen aus SemEval-2010 Task 17 und SemEval-2007 Task 7

Das beste Resultat im SemEval-2010 Task 17 „All-words Word Sense Disambiguation on a Specific Domain“ erreicht das System mit der ID CFILT-2. Das beste Ergebnis im SemEval-2007 Task 7 „Coarse-Grained English All-Words“ erreicht das System NUS-PT. Es handelten sich hierbei jeweils um einen überwachten WSD-Algorithmus. Überwachte WSD-Algorithmen erzielen in den meisten Fälle bessere Ergebnisse als nicht-überwachte WSD-Systeme.

Alle in dieser Arbeit implementierten Algorithmen sind wissensbasiert. Daher kann der beste Algorithmus (der dritte Algorithmus) aus dieser Arbeit nicht mit den überwachten WSD-Algorithmen aus dem SemEval 2017 Task 17 und dem SemEval 2007 Task 7 verglichen werden.

Der bestvergleichbare Algorithmus mit Präzision von 0,512 und Recall 0,495 $\pm$ 0,023 aus dem SemEval 2010 Task 17 ist das System mit der ID CFILT-3, das die 7-te Stelle im Ranking erreicht hat. Dieser Algorithmus verwendete WordNet als Wissensquelle. Der dritte Algorithmus dieser Arbeit hat eine Präzision von 0,4 für das Korpus aus Task 17 erzielt. Dieser Wert lässt sich an der Stelle 22 von den möglichen 29 im Ranking aus dem SemEval 2010 Task 17 einordnen.

Ursache hierfür ist, dass das Bestimmen der Wortart und der Grundform nicht immer richtig war, denn es konnten nicht alle aufzulösenden Wörter mit der richtigen Wortart und Grundform versehen werden. Somit konnten keine richtigen Daten aus der Wissensquelle für die künftigen Schritte der Algorithmen geholt werden. Diese Fehler akkumulieren sich und führen dazu, dass die Generierung der Domain unpräzise wird.

Es soll jedoch noch erwähnt werden, dass die ersten 6 Platzierungen an überwachte Systeme gingen. Die Ergebnisse dieser Arbeit für das Korpus aus Task 7 können nicht mit den Ergebnissen der Systeme aus dem SemEval 2007 Task 7 verglichen werden, da das in dieser Arbeit verwendete Korpus nur einen Text (D004) enthielt, während die Systeme aus dem SemEval 2007 Task 7 ein Korpus disambiguieren mussten, das mit fünf Texten aus unterschiedlichen Domänen besteht.

6 Zusammenfassung

In dieser Arbeit wurden Methoden für „Auflösung von sprachlichen Mehrdeutigkeiten für spezifische Domänen mittels Simulated Annealing“ entwickelt. Die Methoden basieren auf der Überlappung zwischen den Wörtern eines Satzes und ihrer Definitionen aus WordNet untereinander, ebenfalls mit den Schlagwörtern der Domain des Textes.

WSD-Algorithmen für spezifische Domänen sind im Einsatz hilfreicher als WSD-Algorithmen, da die aufzulösenden Texte in den meisten Fällen domain-spezifisch sind. Die Adaption der spezifischen Domain bei nicht-überwachten WSD-Methoden erlaubt es, dass die Systeme robuster werden.

Als Ausgangspunkt wurde ein Simulated Annealing Algorithmus für generelle Domain genommen. Danach wurden Erweiterungen am Algorithmus durchgeführt, sodass die Domain ebenfalls bei der Energieberechnung und bei der Auswahl von den Konfigurationen betrachtet wurde.

Die Rechentechnik von Simulated Annealing erlaubt es, gute Approximationen in machbarer Zeit zu erzielen, denn es werden alle aufzulösenden Wörter aus einem Satz gleichzeitig disambiguiert. Die erreichten Resultate sind mit anderen Ansätzen vergleichbar. Es wurde im Abschnitt Analyse 5.2 dargestellt und nachgewiesen, dass bessere Ergebnisse durch einen verbesserten Algorithmus zur Bestimmung der Wortart und der Grundform erreicht werden können. Somit würde die Domain noch exakter und sehr kontextnah bestimmt werden. Das Feintuning der Parameter wie Temperatur, Abkühlungsfaktor und Anzahl der Iterationen ermöglicht eine verbesserte Performance des Systems.

Literatur

[1] S. Banerjee und T. Pedersen. „An adapted Lesk algorithm for word sense disambiguation using WordNet“. In: *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics* (2002).

[2] Jim Cowie, Joe Guthrie und Louise Guthrie. „Lexical Disambiguation using Simulated Annealing“. In: (1992).

[3] C. Leacock und M. Chodorow. „Combining local context and WordNet similarity for word sense identification“. In: (1998).

[4] M. Lesk. „Automatic sense disambiguation using machine readable dictionaries: how to tell a pine code from an ice cream cone“. In: *Proceedings of the 5th annual international conference on Systems documentation* (1986).

[5] ROBERTO NAVIGLI. „Word Sense Disambiguation: A Survey“. In: (2009).

[6] S. Patwardhan, S. Banerjee und T. Pedersen. „Using measures of semantic relatedness for word sense disambiguation“. In: *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics* (2003).

[7] Ted Pedersen, Satanjeev Banerjee und Siddharth Patwardhan. „Maximizing Semantic Relatedness to Perform Word Sense Disambiguation“. In: (2005).

[8] *Word Sense Disambiguation, Algorithms and Applications)*. Springer, 2006.

[9] *Wordnet: An Electronic Lexical Database (Language, Speech, and Communication)*. Mit Pr, 1998.

[10] Z. Wu und M. Palmer. „Verb semantics and lexical selection, in: 32nd Annual Meeting of the Association for Computational Linguistics, Las Cruces, New Mexico“. In: (1994).