Internetgestützte Textanalyse zur Extraktion von Produktentwicklungswissen mittels OntoUSP: eine Machbarkeitsanalyse

Chen Wang

Studiengang: INFORMATIK

Prüfer: Univ-Prof. Hon-Prof. Dr. Dieter Roller
Betreuer: M. Sc. Julian Eichhoff
begonnen am: 06.11.2012
beendet am: 08.05.2013

CR-Klassifikation: I.2.7 I.5.2 J.6
Inhaltsverzeichnis

1 Einführung .............................................................................................................................................. 7
   1.1 Ziel der Diplomarbeit ................................................................................................................................. 9
   1.2 Problem definieren ......................................................................................................................................... 9

2 Stand der Technik und Grundlagen ........................................................................................ 10
   2.1 Stand der Technik ......................................................................................................................................... 10
   2.2 Grundlagen .................................................................................................................................................. 10
      2.2.1 Stanford Parser ................................................................................................................................... 11
      2.2.2 Semantik Analyse ............................................................................................................................... 13
      2.2.3 Prädikatenlogik erster Stufe (Abk. PL1) ........................................................................................... 14
      2.2.4 Quasi-logische Form, POS Tagging und Lambda(-Kalkül) Notation ................................................. 16
         2.2.4.1 Quasi-logische Form ................................................................................................................. 16
         2.2.4.2 Lambda(-Kalkül) Notation ........................................................................................................ 17
         2.2.4.3 POS Tagging ............................................................................................................................. 17
      2.2.5 Markov Logik Netzwerk (Abk. MLN) ............................................................................................... 17
         2.2.5.1 Markov Netzwerk (Abk. MN) ................................................................................................... 17
         2.2.5.2 Markov Logik Netzwerk (Abk. MLN) ...................................................................................... 18
      2.2.6 Clusteranalyse .................................................................................................................................... 20
      2.2.7 Unsupervised Semantic Parsing und Ontology Unsupervised Semantic Parsing ........................... 20
         2.2.7.1 Unsupervised Semantic Parsing ................................................................................................ 21
         2.2.7.2 Ontology Unsupervised Semantic Parsing ................................................................................ 26

3 Entwurf ..................................................................................................................................................... 28
   3.1 Ausgabedatei für USP .................................................................................................................................. 29
   3.2 Der erste Ansatz : Generierung der Regeln für Informationsextraktion .................................................. 31
   3.3 Der zweite Ansatz : Generierung der Hierarchie der Wörter ................................................................... 33

4 Implementierung ....................................................................................................................................... 35
   4.1 Generierung der Hierarchie durch Bestimmung der Abhängigkeit von den Wörtern .............................. 35
   4.2 Vorverarbeitung ......................................................................................................................................... 35
      4.2.1 Verschiedene Fälle ............................................................................................................................. 35
      4.2.2 Bestimmung der Abhängigkeit ........................................................................................................ 36
      4.2.3 Bestimmung von Produkteigenschaften und Produkteigenschaftswerten ...................................... 38
      4.2.4 Aktualisierung der Werte von ClusterSum, Count und CClusterSum ............................................. 38
   4.3 Bearbeitung der Knoten mit CC>=0.5 ........................................................................................................ 40
      4.3.1 Vier Fälle ............................................................................................................................................ 40
         4.3.1.1 Der erste Fall ............................................................................................................................. 40
         4.3.1.2 Der zweite Fall .......................................................................................................................... 41
         4.3.1.3 Der dritte Fall ............................................................................................................................ 42
         4.3.1.4 Der vierte Fall ........................................................................................................................... 43
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.2</td>
<td>Aktualisierung der Werte von ClusterSum, Count und CClusterSum</td>
<td>44</td>
</tr>
<tr>
<td>4.4</td>
<td>Verarbeitung der Konten mit CC&lt;0.5</td>
<td>44</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Bestimmung der Abhängigkeit</td>
<td>44</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Aktualisierung der Werte von ClusterSum, Count und CClusterSum</td>
<td>45</td>
</tr>
<tr>
<td>4.5</td>
<td>Verbessern die Hierarchie</td>
<td>46</td>
</tr>
<tr>
<td>4.6</td>
<td>Filter</td>
<td>48</td>
</tr>
<tr>
<td>4.7</td>
<td>Ausgaben Filter</td>
<td>50</td>
</tr>
<tr>
<td>5</td>
<td>Experiment</td>
<td>51</td>
</tr>
<tr>
<td>6</td>
<td>Zusammenfassung</td>
<td>56</td>
</tr>
<tr>
<td>7</td>
<td>Ausblick</td>
<td>57</td>
</tr>
<tr>
<td>7.1</td>
<td>Nicht löschbare Probleme</td>
<td>57</td>
</tr>
<tr>
<td>7.2</td>
<td>löschbare Probleme</td>
<td>58</td>
</tr>
<tr>
<td>7.3</td>
<td>Arbeit für die Zukunft</td>
<td>58</td>
</tr>
<tr>
<td>7.4</td>
<td>Begrenzung der System-Anforderung</td>
<td>59</td>
</tr>
<tr>
<td>8</td>
<td>Literaturverzeichnis</td>
<td>60</td>
</tr>
</tbody>
</table>
Abbildungsverzeichnis

Abbildung 1: Der Ablauf von Data Mining................................................................. 7
Abbildung 2: Ablauf der Sprachanalyse................................................................. 8
Abbildung 3: Der Durchlauf der Informationsextraktion........................................ 11
Abbildung 4: Der Parser Baum, der von Stanford Parser erzeugt wird................. 12
Abbildung 5: Der Parser Baum, der manuell erzeugt wird................................. 13
Abbildung 6: Ein Beispiel für Shallow Semantic Parsing.................................... 13
Abbildung 7: Ein Beispiel für semantische Analyse............................................. 14
Abbildung 8: (aus [17]) Beispiele für Wissensdomäne von Prädikatenlogik erster Stufe. Fr() ist Abkürzung für Friends(), Sm() für Smokes(), and Ca() für Cancer()....................................................... 16
Abbildung 9 (aus [19]): Die Umwandlung des Satzes „Everybody speaks two languages“ in QLF und entsprechender Prädikatenlogik.................................................................................. 16
Abbildung 10: Ein Satz markiert mit POS Tag...................................................... 17
Abbildung 11: (aus [17]): Der MLN Graph, der Tabelle 2 entspricht. A und B sind die Konstante........................................................... 19
Abbildung 12 (aus [23]): Cliques und Gewichte von MLN in Abbildung 11. Es gibt 6 Cliques.............................................................. 19
Abbildung 14 (aus [24]): Illustration für Clustering-Verfahren von USP.............. 21
Abbildung 15: Beispiel für die Generierung von QLF............................................ 22
Abbildung 16: Die Partitionen von QLF. Wenn die Atomen die gleichen Bedeutungseinheit besitzen, werden die Atomen in einem Cluster bzw. in einer Partition hinzufügt.................................................. 23
Abbildung 17: Die sub-Formeln von QLF .............................................................. 23
Abbildung 18: Ein Beispiel für Erzeugung der Lambdaformen................................ 24
Abbildung 19: Die Lambdaformen werden auf den Cluster aufgeteilt und den syntaktischen Variationen in Argumenten zugeordnet. Links sind die Lambdaformen und rechts sind die Cluster ................... 24
Abbildung 20: Beispiel für ein Cluster................................................................. 25
Abbildung 21: Beispiel für QLF Partition. Form(p, f!), ArgForm(p, i, f!) sind QLF Partitionen................................................................. 25
Abbildung 22: Beispiel für ArgType(p, i, a!), Arg(p, i, p!), Number(p, a, n)......... 26
Abbildung 23: Beispiel für Objekt und Eigenschaft Cluster. In „property cluster“ sind Argumentformen, Argumente von Core Formen sind Argument-Numbers. .................................................... 27
Abbildung 24: Architektur für Extraktion von Produkten, Produkteigenschaften und Produkteigenschaftswerten......................................................... 28
Abbildung 25: Ein Beispiel für eine MLN Datei.................................................... 29
Abbildung 26: Ein Beispiel für eine PARSE Datei................................................. 30
Abbildung 27: Beispiel für die graphische Darstellung von „*.mln“ Datei............. 31
Abbildung 28: links ist die graphische Darstellung von Markov Logik Netzwerk. Rechts ist die gerichtete graphische Darstellung, entspricht der graphische Darstellung von MLN. ..................................................... 33
Abbildung 29: Ein weiteres Beispiel für graphische Darstellung von „*.mln“ Datei ................................................................. 33
Abbildung 30: Illustration für einige Definitionen................................................. 35
Abbildung 31: Ein Teil von „*.mln“........................................................................ 36
Abbildung 32: Oben ist die graphische Darstellung von Abbildung 31. Nach der Bestimmung der Abhängigkeiten (CC=1) wird die graphische Darstellung (unten) erzeugt.................................................. 37
Abbildung 33: Es gibt die Abhängigkeit zwischen Knoten „ips” und „display“, aber die Abhängigkeit ist nicht deutlich ........................................................................................................... 37
Abbildung 34: Beispiel für die Darstellung von Blättern in „*.mln“, 31 ist ein Blatt und 30 ist kein Blatt.......................... 38
Tabellenverzeichnis

Tabelle 1: Beispiel für Stanford Parser ...................................................................................................................... 12
Tabelle 2: (aus 6) Beispiele für Wissensdomäne von MLN....................................................................................... 19
1 Einführung


Abbildung 1: Der Ablauf von Data Mining


Abbildung 2: Ablauf der Sprachanalyse


8

1.1 Ziel der Diplomarbeit

In dieser Diplomarbeit wird geprüft, ob OntoUSP eine Methode ist, mit der das Produkt, die Produkteigenschaft und die Werte der Produkteigenschaft aus einem natürlichen Text extrahiert werden können. Zwar ist das Programm von OntoUSP nicht vorhanden, aber man kann von den Ergebnissen von USP ausgehen, weil OntoUSP eine Erweiterung von USP ist, und bei Experimenten mit Onto USP werden die Ausgaben von USP benutzt, d.h. die Ausgaben von OntoUSP enthalten die gleichen oder ähnliche Informationen wie USP.

1.2 Problem definieren


- Die Wurzel ist entweder ein Produkt oder ein Markenzeichen
- Die Blätter sind die Werte der Produkteigenschaften
- Die Eltern Knoten der Blätter sind Produkteigenschaften
2 Stand der Technik und Grundlagen

2.1 Stand der Technik


USP liefert vollständig semantische Analysen und ein Markov Logik Netzwerk für Wörter. Damit werden die o.g. Schwächen überwunden.

2.2 Grundlagen

Ausgaben von USP sind "*.parse" und "*.mln" Dateien, wobei die "*.parse" eine Baumstruktur liefert, und "mln" ein Netz der Wörter liefert. Durch "*.parse" und "*.mln" Dateien werden die Ausgaben generiert.

Abbildung 3: Der Durchlauf der Informationsextraktion

Im Folgend werden die Grundlagen für USP System sowie die Komponente für USP System, Grundlagen für Logik, Grundlagen für Markov Logik Netzwerk und Grundlagen für Linguistik besprochen.

2.2.1 Stanford Parser

"Der Stanford Parser ist ein probabilistisches Parser Programm. Die natürliche Sprache ist die Eingabe für das Programm und durch den probabilistischen Parser kann die grammatische Struktur der Sätze bestimmt werden. Die Kenntnisse der Wahrscheinlichkeiten von probabilistischen Parseern werden aus manuellen analysierten Sätzen erworben, und mithilfe der probabilistischen Parser können die wahrscheinlichste Analyse von neuen Sätzen produziert werden. Dieser probabilistischen Parser garantiert nicht, dass die Ergebnisse 100% richtig sind."

markiert. Die syntaktische Struktur der Nominalphrase aus Stanford Parser ist NP→NP PP_with NP, die richtige ist NP→NP CONJ_and NP. Solche Fehler führen zu einer falschen Ontologie.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Apple announces 13 MacBook Pro with Retina display and new iMac.</td>
</tr>
<tr>
<td>2</td>
<td>nsubj(announce-2, Apple-1), num(pro-5, 13-3), nn(pro-5, MacBook-4), dobj(announce-2, pro-5), nn(display-8, Retina-7), prep_with(announce-2, display-8), amod(imac-11, new-10), conj_and(display-8, imac-11)</td>
</tr>
<tr>
<td>3</td>
<td>Apple_NNP, announces_VBZ, 13_CD, MacBook_NNP, Pro_NNS, with_IN, Retina_NNP, display_NN, and_CC, new JJ, iMac_NN, _ _</td>
</tr>
<tr>
<td>4</td>
<td>Apple, announce, 13, MacBook, pro, with, Retina, display, and, new, imac, _ _</td>
</tr>
</tbody>
</table>

Tabelle 1: Beispiel für Stanford Parser


Abbildung 4: Der Parser Baum, der von Stanford Parser erzeugt wird.
2.2.2 Semantik Analyse

Für die Verarbeitung natürlicher Sprache gibt es folgende Schwierigkeiten: die heterogenen Wissensdomänen, die Auflösung der Ambiguität der Sprache, Modellierung der Sprache und die Auflösung der syntaktischen Variationen der Sprache. Im Durchlauf von „Shallow Semantic Parsing“ erfolgt die Erkennung von semantischer Repräsentation z.B. „Wer“, „Was“, „Wann“, „Wo“, „Warum“, „Wie“, usw. und nur die Elemente, die aufeinander folgend und zusammenhängend sind, werden ermittelt. „Shallow Semantic Parsing“ kann die s.g. Schwierigkeiten nicht auflösen und es fehlt die Fähigkeit der Schlussfolgerung.

\[
\text{Tom loves Mary.} \quad \xrightarrow{\text{Shallow Semantic Parsing}} \quad [\text{AGENT Tom}] \text{ loves } [\text{RECEPIENT Mary}].
\]

Im Gegensatz zu „Shallow Semantic Parsing“ liefert die vollständige semantische Analyse eine Repräsentation eines Satzes in Prädikatenlogik erster Stufe oder andere formale Sprache und unterstützt eine automatische Schlussfolgerung. Die natürliche Sprache wird im Leseprozess vollständig syntaktisch analysiert, damit die logischen Formen erzeugt werden können. Durch semantische Analyse wird die von syntaktischer Analyse erzeugte logische Form, nämlich die logische Repräsentation der natürlichen Sprache, auf der vollständigen semantischen Repräsentation, nämlich die Bedeutungsrepräsentation, abgebildet. Die Bedeutungsrepräsentationen der Sprache werden in dieser Ausarbeitung durch Prädikatenlogik erster Stufe dargestellt. Einige Definitionen müssen hier angeben werden:

- Ein Term ist ein Objekt in einer Domäne. Ein Term kann eine Konstante, eine Variable oder eine Funktion, die auf den Variablen angewendet, sein.
Eine Formel bzw. eine atomare Formel ist ein Prädikatsymbol, das auf n-Tupel von Termen angewendet wird. Eine Formel kann aus mehreren atomaren Formeln verknüpft mit logischen Symbolen und Quantoren rekursiv konstruiert sein.

Unter einer lexikalischen Einheit kann in dieser Ausarbeitung ein Wort verstanden werden. Ein lexikalischer Eintrag definiert die logische Formel für eine lexikalische Einheit mit POS Tagging. λ-gebundene Variablen markieren die fehlenden Argumenten in den logischen Formen.


2.2.3 Prädikatenlogik erster Stufe (Abk. PL1)


Einige Definitionen (aus [16]):

- Eine Variable hat die Form $x_i$
- Ein Prädikatsymbol hat die Form $P_i$ und ein Funktionssymbol hat die Form $f_i$
- Jede Variable ist ein Term, jede Konstante ein Term, sowie $f (t_1, \ldots, t_n)$ auch ein Term, falls $f$ eine Funktion und $t_i$ die Terme sind.
- $P(t_1, \ldots, t_k)$ ist eine Formel bzw. eine atomare Formel, fall $P$ ein Prädikatsymbol ist und $t_i$ Terme sind.

Abbildung 7: Ein Beispiel für semantische Analyse

Die ersten drei Zeilen in der Abbildung 7 sind lexikalische Einträge bzw. Wörter. Die syntaktische Kategorie bzw. POS Tagging von „love“ ist „Verb“. Wenn zwei Atome die Funktion „loves(x,y)“ erfüllen, dann ist diese Funktion true. Die letzten zwei Zeilen haben gezeigt, dass die lexikalischen Einträge in einem größeren Fragment der Bedeutung zusammengestellt werden.

Verb[$\lambda y \lambda x.l Olivia(x,y)] \rightarrow loves$
NP[Tom] $\rightarrow$ Tom
NP[Mary] $\rightarrow$ Mary
VP[rel(obj)] $\rightarrow$ Verb[rel] NP[obj]
S[rel(obj)] $\rightarrow$ NP[obj] VP[rel]
Für jede Formel \( \neg F \), \( \exists F \), \( \forall F \), \( F \lor G \), \( F \land G \) sind auch die Formeln, wobei \( \exists \) und \( \forall \) sind die Quantoren.

Alle vorkommenden Variablen sind entweder frei oder gebunden. Wenn \( x \) in der Form \( \exists xF \) oder \( \forall xF \) vorkommt, dann heißt die Variable \( x \) in Formel \( F \) gebunden, andernfalls heißt frei.


Einige Definitionen (aus [17]):

- Jede Konstante ist ein Objekt in einer Wissensdomäne und kann typisiert sein, z.B. Konstant HA repräsentiert Hersteller Apple.
- Die Variablen können typisiert sein und repräsentieren die Objekte gleichen Typs in einer Wissensbasis, z.B. Variable \( x_i = \text{Tom} \) ist der Name der Menschen in der Wissensbasis „MenschenName (Tom, Jerry, Mary)“. Durch die Substitution der Variablen durch eine Konstante aus der Konstante Menge werden die verschiedenen Objekte entstehen, z.B. MenschenName (Tom) und MenschenName (Jerry).
- Die Beziehung zwischen den Objekten sind die Funktionen wie Mutter_von, guter_Freund_von.
- Ein Prädikatsymbol repräsentiert die Beziehung zwischen den Objekten und die Eigenschaften der Objekte, z.B. Feind und Rauchen.
- Ein Term kann eine Konstante, eine Variable oder eine Funktion sein, die auf Unterterme angewendet werden kann.
- Eine atomare Formel ist ein Prädikatsymbol, das auf Unterterme angewendet wird.
- Eine Formel kann rekursiv aus atomaren Formeln, die mit Quantoren (\( \exists \), \( \forall \)) und logischen Symbolen (\( \land \), \( \lor \), \( \leftrightarrow \), \( \Rightarrow \)) der Prädikatenlogik verküpft sind, konstruiert werden.
- Ein Grundterm ist ein Term, der keine Variable enthält.
- Ein Grundatom oder ein Grundprädikat ist eine atomare Formel, deren Argumente alle Grundterme sind.

Abbildung 8: (aus [17]) Beispiele für Wissensdomäne von Prädikatenlogik erster Stufe. Fr() ist Abkürzung für Friends(), Sm() für Smokes(), and Ca() für Cancer().

2.2.4 Quasi-logische Form, POS Tagging und Lambda(-Kalkül) Notation

2.2.4.1 Quasi-logische Form


a. *Everybody speaks two languages*

<table>
<thead>
<tr>
<th>English</th>
<th>First-Order Logic</th>
<th>Clausal Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>Friends of friends are friends.</td>
<td>$\forall x \forall y \forall z \text{Fr}(x, y) \land \text{Fr}(y, z) \Rightarrow \text{Fr}(x, z)$</td>
<td>$\neg \text{Fr}(x, y) \lor \neg \text{Fr}(y, z) \lor \text{Fr}(x, z)$</td>
</tr>
<tr>
<td>Friendless people smoke.</td>
<td>$\forall x (\neg (\exists y \text{Fr}(x, y)) \Rightarrow \text{Sm}(x))$</td>
<td>$\text{Fr}(x, g(x)) \lor \text{Sm}(x)$</td>
</tr>
<tr>
<td>Smoking causes cancer.</td>
<td>$\forall x \text{Sm}(x) \Rightarrow \text{Ca}(x)$</td>
<td>$\neg \text{Sm}(x) \lor \text{Ca}(x)$</td>
</tr>
<tr>
<td>If two people are friends, either both smoke or neither does.</td>
<td>$\forall x \forall y \text{Fr}(x, y) \Rightarrow (\text{Sm}(x) \Leftrightarrow \text{Sm}(y))$</td>
<td>$\neg \text{Fr}(x, y) \lor \text{Sm}(x) \lor \neg \text{Sm}(y), \lor \text{Fr}(x, y) \lor \neg \text{Sm}(x) \lor \text{Sm}(y)$</td>
</tr>
</tbody>
</table>

Abbildung 9 (aus [19]): Die Umwandlung des Satzes „Everybody speaks two languages“ in QLF und entsprechender Prädikatenlogik.
2.2.4.2 Lambda(-Kalkül) Notation

„Durch Lambda(λ)-Ausdrücke werden formale Parameter eingeführt, die durch Terme ersetzt werden können.“ [20] Beide \(\lambda x.\text{Love}(x, y)\) und \(\lambda x.\lambda y.\text{Love}(x, y)\) sind die Lambda-Ausdrücke. Die \(\lambda\)-gebundene Variable kann durch ein Argument aus einem Definitionsbereich, z.B. aus konstanten Menge in MLN, substituiert werden. Die nicht-\(\lambda\)-gebundene Variable heißt frei. Ein Beispiel für Substitution einer \(\lambda\)-gebundenen Variable:

\[\lambda x.\text{Love}(x, y)(\text{Tom}) \Rightarrow \text{Love}(\text{Tom}, y)\]

wobei \(y\) eine nicht-\(\lambda\)-gebundene Variable bzw. eine freie Variable ist. Die Substitution von \(\lambda\)-gebundenen Variable ist die \(\lambda\)-Reduktion. „Ein Prädikat mit mehreren Argumenten kann durch die \(\lambda\)-Reduktion auf eine Folge von jeweils einstelligen Prädikaten abbilden.“ [20]

2.2.4.3 POS Tagging


\[\text{Tom loves Mary.}\]

\[(S(NP(NNP Tom)) (VP(VBZ loves)(NP (NNP Mary)))\]

Abbildung 10: Ein Satz markiert mit POS Tag.

2.2.5 Markov Logik Netzwerk (Abk. MLN)

2.2.5.1 Markov Netzwerk (Abk. MN)

Das Folgende baut auf [17] auf. „Markov Netzwerk (oder Markov Random Field) ist ein statistisches Modell für multivariate Verteilung einer Menge von Variablen \(X = (X_1, X_2, \ldots, X_n) \in \mathbb{R}\) und beschreibt die ungerichteten Graphen, die bedingte Unabhängigkeitsaussagen zwischen Variablen ausdrücken.“ [17] & [21] In einem ungerichteten Graph repräsentiert jeder Knoten eine Variable, jede Clique im Graph hat eine potenzielle Funktion, die einen Zustand der Clique repräsentiert. Die multivariate Verteilung von MN:

\[P(X = x) = \frac{1}{Z} \Pi \Phi_k(x_{\{k\}}) \quad (1)\]

wobei \(x_{\{k\}}\) ist der Zustand von k-ste Clique ist, d.h. \(x_{\{k\}}\) repräsentieren alle Werte von den Variablen in der k-ste Clique. \(Z\) ist die Normalisierung. Die Formel (1) kann als log-lineares Modell dargestellt werden.
\[ P(X = x) = \frac{1}{Z} \exp \left\{ \sum_j \omega_j f_j(x) \right\} \quad (2) \]

wobei \( \omega_j \) Gewicht von einer Feature Funktion \( f_j(x) \) ist. In dieser Ausarbeitung \( f_j(x) \in \{0,1\} \).

Feature Funktion beschreibt einen Zustand der Clique und das Gewicht der Feature Funktion ist 
\[ \log \phi_k(x_{\{k\}}). \]

### 2.2.5.2 Markov Logik Netzwerk (Abk. MLN)


Einige Definition (aus [17]):

- Ein Markov Logik Netzwerk \( L \) ist eine Menge von Paaren \((F_i, \omega_i)\), wobei \( F_i \) eine logische Formel der Prädikatenlogik der erster Stufe ist und \( \omega_i \) ein Gewicht. Hier werden nur die existenzquantifizierten Variablen betrachtet und alle allquantifizierten Variablen sind die freien Variablen.
- \((F_i, \omega_i)\) zusammen mit einer endlichen Menge von logischen Konstanten \( C = \{c_1, c_2, \ldots, c_{|C|}\} \) definieren ein Markov Logik Netz \( M_{L,C} \).
- Jedes Grundatom in \( L \) entspricht einem binären Wert Knoten in \( M_{L,C} \). Die Grundatome Menge \( X = \{X_1, \ldots, X_n\} \) wird dadurch erhalten, dass die Variablen der prädikatenlogischen Formel in \( L \) durch die in \( L \) gegebenen Konstanten substituiert werden. Der Wert eines Knotens ist genau dann 1, wenn das Grundatom wahr ist, ansonsten ist der Wert 0. Der Knoten kann mit anderen Knoten durch die Kanten verbunden werden, wenn die beiden Knoten bzw. die Grundatome in einer Belegung der Grundformel gemeinsam vorkommen.
- Für jede Belegung einer Grundformel \( F_i \) in \( L \) besitzt ein Feature \( f_i \), der Wert von \( f_i \) genau dann 1 ist, wenn die Belegung der Grundformel wahr ist und sonst 0. Die Summe der Gewichte für Feature \( f_i \) ist das Gewicht \( w_i \) in \( L \).
Tabelle 2: (aus [17]) Beispiele für Wissensdomäne von MLN.

<table>
<thead>
<tr>
<th>English</th>
<th>First-Order Logic</th>
<th>Clausal Form</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Friends of friends are friends.</td>
<td>$\forall x \forall y \forall z \text{Fr}(x, y) \land \text{Fr}(y, z) \Rightarrow \text{Fr}(x, z)$</td>
<td>$\neg \text{Fr}(x, y) \lor \neg \text{Fr}(y, z) \lor \text{Fr}(x, z)$</td>
<td>0.7</td>
</tr>
<tr>
<td>Friendless people smoke.</td>
<td>$\forall x \left( \neg \left( \exists y \text{Fr}(x, y) \right) \Rightarrow \text{Sm}(x) \right)$</td>
<td>$\text{Fr}(x, y) \lor \text{Sm}(x)$</td>
<td>2.3</td>
</tr>
<tr>
<td>Smoking causes cancer.</td>
<td>$\forall x \text{Sm}(x) \Rightarrow \text{Ca}(x)$</td>
<td>$\neg \text{Sm}(x) \lor \text{Ca}(x)$</td>
<td>1.5</td>
</tr>
<tr>
<td>If two people are friends, either both smoke or neither does.</td>
<td>$\forall x \forall y \text{Fr}(x, y) \Rightarrow (\text{Sm}(x) \leftrightarrow \text{Sm}(y))$</td>
<td>$\neg \text{Fr}(x, y) \lor \text{Sm}(x) \lor \neg \text{Sm}(y)$</td>
<td>1.1</td>
</tr>
</tbody>
</table>

In der Tabelle 2 ist Fr( ) Abkürzung für Friends( ), Sm( ) für Smokes( ), and Ca( ) für Cancer( ). Im Vergleich zu Tabelle 2 werden hier die Gewichte hinzugefügt. Allen Regeln in dem Spalt „First-Order Logic“ sind die Grundformeln und die Belegungen der Grundformeln in dem Spalt „Clausal Form“. Wenn eine Grundformel mehrere Belegungen besitzt, dann wird das Gewicht der Grundformel gleichmäßig auf Belegungen aufgeteilt, z.B. Das Gewicht für $\forall x \forall y \text{Fr}(x, y) \Rightarrow (\text{Sm}(x) \leftrightarrow \text{Sm}(y))$ ist 2.2 und das Gewichte für jede Belegung ist 1.1.

Abbildung 11: (aus [17]) : Der MLN Graph, der Tabelle 2 entspricht. A und B sind die Konstant.

<table>
<thead>
<tr>
<th>First-Order logic</th>
<th>Variable assignment</th>
<th>Clique</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_1: \forall x, \text{Sm}(x) \Rightarrow \text{Ca}(x)$</td>
<td>$x=A$</td>
<td>${\text{Sm}(A), \text{Ca}(A)}$</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>$x=B$</td>
<td>${\text{Sm}(B), \text{Ca}(B)}$</td>
<td>1.5</td>
</tr>
<tr>
<td>$F_2: \forall x \forall y, \text{Fr}(x, y) \Rightarrow (\text{Sm}(x) \leftrightarrow \text{Sm}(y))$</td>
<td>$x=A, y=A$</td>
<td>${\text{Fr}(A, A), \text{Sm}(A)}$</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td>$x=A, y=B$</td>
<td>${\text{Fr}(A, B), \text{Sm}(A), \text{Sm}(B)}$</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td>$x=B, y=A$</td>
<td>${\text{Fr}(B, A), \text{Sm}(A), \text{Sm}(B)}$</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td>$x=B, y=B$</td>
<td>${\text{Fr}(B, B), \text{Sm}(B)}$</td>
<td>1.1</td>
</tr>
</tbody>
</table>


Aus der Definition von MLN wird jeder Knoten in MLN $M_{L,C}$ durch Einsetzen für Variablen der logischen Formeln in MLN die Grundatome erzeugt. Die Kante zwischen den Knoten entspricht die Beziehung zwischen den Knoten. Deshalb kann MLN als Model von Markov Netzwerk gesehen werden und die Wahrscheinlichkeitsverteilung ist:

$$P(X = x) = \frac{1}{z} \exp(\sum_i \omega_i n_i(\chi)) = \frac{1}{z} \prod_i \phi_i(x_{(i)})^{n_i(x)}$$ (5)  

19
wobei \( n_i \) die Anzahl der Grundformeln ist, die den Wert 1 haben. Formel (5) hat gezeigt, dass eine Welt nicht unmöglich ist, wenn diese Welt eine Randbedingung oder mehrere Randbedingungen verletzt, sondern besitzt geringere Wahrscheinlichkeit. Gewicht \( \omega_i \) zeigt, wie „stark“ die Randbedingung in der Welt ist. Für ein bestimmtes MLN können unterschiedliche Markov Netzwerk erzeugt werden, wenn die Konstant Menge unterschiedlich sind, aber diese Markov Netzwerk haben auch manche Gemeinsamkeiten wie die gleiche Anzahl der Clique. Die Gewichte werden entweder aus dem Lernen von Trainingsdaten erhalten oder von Menschen manuell gegeben.

### 2.2.6 Clusteranalyse


![Clusteranalyse Beispiel](image)


### 2.2.7 Unsupervised Semantic Parsing und Ontology Unsupervised Semantic Parsing

Früher wurde der semantische Parser manuell erstellt, zwar einige Ansätze für maschinelles Lernen wurden danach entwickelt, aber die manuelle Unterstützung war immer noch erforderlich, und manche Ansätze beschränkten sich auf einer geschlossenen Wissensdomäne. USP ist der erste nicht-überwachte maschinelles Lernen Ansatz für Semantik Parser. Ob ein
Ansatz für maschinelles Lernen überwacht ist, ist abhängig davon, ob Eingabe- und Ausgabedatei manuell markiert sind.

- **nicht-überwachtes Lernen (engl. unsupervised learning)**: Der Algorithmus erzeugt für eine gegebene Menge von Eingaben ein Modell, das die Eingaben beschreibt und Vorhersagen ermöglicht. Dabei gibt es Clustering-Verfahren, die die Daten in mehrere Kategorien einteilen, die sich durch charakteristische Muster voneinander unterscheiden.

### 2.2.7.1 Unsupervised Semantic Parsing

USP beruht auf drei zentralen Ideen: (aus [7])

- Die gleiche Formen können clusterd werden. Die Formen, die aus den gleichen Formen bestehen, können clusterd werden.

**USP = Recursively cluster expressions with similar subexpressions**

- Microsoft buys Powerset
- Microsoft acquires semantic search engine Powerset
- Powerset is acquired by Microsoft Corporation
- The Redmond software giant buys Powerset
- Microsoft’s purchase of Powerset, ...

**Cluster same forms at the atom level**

**USP = Recursively cluster expressions with similar subexpressions**

- Microsoft buys Powerset
- Microsoft [acquires] semantic search engine Powerset
- Powerset is acquired by Microsoft Corporation
- The Redmond software giant buys Powerset
- Microsoft’s purchase of Powerset, ...

**Cluster forms in composition with same forms**

Abbildung 14 (aus [24]): Illustration für Clustering-Verfahren von USP

- USP startet direkt von syntaktischen Analysen und konzentriert sich nur auf deren Umsetzung zum semantischen Inhalt. Die vorherige entwickelte Parser können in USP eingesetzt werden, deshalb stehen viele Ressourcen zur Verfügung. Die syntaktische Analyse und die semantische Analyse sind in USP getrennt, damit die Komplexität der semantischen Analyse reduziert wird, weil es nicht erforderlich ist, bei Zusammensetzung der Bedeutungen ein domänenspezifisches Verfahren zur Erzeugung von Kandidaten Lexikon zu brauchen.

sind die Knoten die Wörter, und die Kanten sind die Beziehungen zwischen den Wörtern. Die QLF in dieser Ausarbeitung wird daraus erhalten:

- jeder Knoten in einem Dependenzbaum wird in ein unäres Atom mit dem Prädikat umwandelt, und das Prädikat wird mit POS Tag markiert, z.B. „Microsoft“ wird in „Microsoft(n4)“ umgewandelt.
- jede Kante in einem Dependenzbaum entspricht einem Prädikat mit zwei Atomen, und das Prädikat ist die „Beziehung“ zwischen den Atomen, z.B. nsubj(n3, n2).

wobei $n_2$ und $n_3$ die Skolemkonstante sind. Eine Skolemkonstante kann man in dieser Ausarbeitung als eine Konstante aus der konstanten Menge in MLN verstehen. (Mehr über Skolemkonstante sieht Kapitel 9 in [25]).


Abbildung 15: Beispiel für die Generierung von QLF.

Die natürliche Sprache wird durch s.g. Stanford Parser drei Dateien als Eingaben für USP System erzeugt. Diese drei Dateien stellen einen Dependenzbaum dar. Der Dependenzbaum
wird in QLF umgewandelt. QLF in dieser Ausarbeitung wird vereinfacht. Deshalb ist die Darstellung hier nicht gleich wie die Darstellung in der Abbildung 9.

In dem USP System wird QLF in den kleinen Teilen partitioniert, und die partitionierte Teile der QLF werden in einer Gruppe bzw. in einem Cluster zugewiesen, wenn sie die gleiche Bedeutung besitzen, z.B. „Microsoft“ und „Corporation“ besitzen die gleiche Bedeutung.

Abbildung 16: Die Partitionen von QLF. Wenn die Atomen die gleichen Bedeutungseinheit besitzt, werden die Atomen in einem Cluster bzw. in einer Partition hinzufügt.

Abbildung 17: Die sub-Formeln von QLF

Manche Atome sind die Bedeutungseinheiten, und mache Atome sind die Argumente. Z.B. : 

\[ \text{buys}(n3) \land \text{nsubj}(n3, n2) \land \text{dobj}(n3, n4) \]

\[ \lambda x_2. \lambda x_4. \text{buys}(n3) \land \text{nsubj}(n3, x_2) \land \text{dobj}(n3, x_4) \]

wobei \( n_1, n_2 \) und \( n_3 \) die Skolemkonstante sind.

Die Lambdaform wird mit Hilfe von Davidsonian Semantics weiter zerlegt in Core Form und Argumentform. Die Core Form ist eine Lambdaform, die keine Lambdavariable enthält, und eine Argumentform ist eine Lambdaform, die nur eine Lambdavariable enthält.
Abbildung 18: Ein Beispiel für Erzeugung der Lambdaformen

Durch die Clusteranalyse werden die Lambdaformen in den Lambda-Form Cluster aufgeteilt. Ein Lambda-Form Cluster ist ein Cluster, das die semantisch austauschbaren Lambdaformen enthält und die Bedeutung der Sub-Formel von QLF. Lambdaform Cluster kann den Argumenttypen enthalten, damit die Typen der Argumente in den Beziehungen unterschieden werden können. Z.B.: die Argumente „ACQUIRER“ und „ACQUIRED“ in den Relationen nsubj(n1, n2) und dobj(n1, n3) entsprechen dem Subjekt und Objekt von Verb „buys“. In Stanford Parser kann die Argument „ACQUIRED“ als „nsubjpass“ für ein Subjekt in einen passiven Satz repräsentiert werden. Die syntaktischen Variationen werden in dem Lambda-Form Cluster abgezogen und unterschieden sich durch den Argumenttypen.


Im USP System startet die semantische Analyse mit Clusteranalyse der Lambda-Formen in Token bzw. in Atom Ebene, d.h. die QLFs werden durch die Partition auf den Sub-Formen der QLF abgebildet, jede Sub-Form hat eine entsprechende Lambdaform, die Lambdaform wird weiter in Core Form und Argumentform zerlegt. Um die Argumentform zu unterscheiden, wird jede Argumentform einem Argumenttyp zugewiesen. Eine Regel, bei der eine Lambda-Form auf ein Cluster abgebildet wird und einen Argumenttyp zuweist, ist eine semantische Grammatik. Mit der semantischen Grammatik werden dann die Core Formen auf die Cluster und die
Argumentformen auf den Argumenttypen abgebildet. USP eine Wahrscheinlichkeitsverteilung über den semantischen Parser. Das Problem von maschinellem Lernen in USP ist das Lernen von s.g. semantischer Grammatik. Das Lernen in USP wird realisiert durch die Nutzung von Markov Logik Netzwerk.

Abbildung 20 : Beispiel für ein Cluster

Ein semantischer Parser L partitioniert ein QLF in die QLF-Teile p1, p2, … , pn, jedes Teil p wird in einen oder einige Lambdaform Cluster c zugewiesen, und pi wird später in Core Form f und Argumentformen f1, f2, … , fk umgewandelt, jede Argumentform besitzt auch einen Argumenttyp a in c. Um die Verteilung über die Lambdaformen zu modellieren, werden Form(p, f!), ArgForm(p, i, f!) definiert, wobei p eine Partition ist, i der Index eines Arguments und f eine Sub-Formel von QLF. Form(p, f!) ist true genau dann, wenn Partition p eine Core Form f hat, und ArgForm(p, i, f!) ist true genau dann, wenn i-stes Argument in p die Sub-Form hat. „f!” Notation bedeutet, dass jede Partition oder jedes Argument nur eine Form hat.

Abbildung 21 : Beispiel für QLF Partition. Form(p, f!), ArgForm(p, i, f!) sind QLF Partitionen.
Die o.g. semantische Grammatik Regeln von semantischer Analyse in USP werden durch folgende 4 Formeln definiert:

\[
\begin{align*}
    p & \in +c \land \text{Form}(p, +f) \\
    \text{ArgType}(p, i, +a) & \land \text{ArgForm}(p, i, +f) \\
    \text{Arg}(p, i, p') & \land \text{ArgType}(p, i, +a) \land p' \in +c' \\
    \text{Number}(p, +a, +n)
\end{align*}
\]

2.2.7.2 Ontology Unsupervised Semantic Parsing

Eine Schwachheit des USP Systems ist das „Sparse Data“, das „Sparse Data“ bezeichnet hier die Information, die selten im Korpus vorkommt. „Sparse Data“ führt zu niedriger Genauigkeit, weil nicht genug Daten zur Verfügung stehen, um die Wahrscheinlichkeit des Ereignisses genau abzuschätzen.“ [26] Onto USP ist die Erweiterung von USP und hat die Fähigkeit, die Informationen zu strukturieren. Im Vergleich zu USP Onto USP führt eine Hierarchie Clusteranalyse durch. Onto USP löst das Problem über das „Sparse Data“ durch das Hierarchie Clusteranalyse.


\[
\begin{align*}
    x & \in +p \land \text{HasValue}(x, +v) \\
    e & \in c \land \text{SubExpr}(x, e) \land x \in p \Rightarrow \exists i. \text{IsPart}(c, i, p)
\end{align*}
\]

wobei:

- \(\text{HasValue}(s, v)\) : Sub-Form der Lambdaform hat dem Wert \(v\).
- \(e \in c\) : in Cluster \(c\) gibt es Lambdaform \(e\).
- \(\text{SubExpr}(s, e)\) : \(s\) ist eine Sub-Form einer Lambdaform.
- \(\text{IsPart}(c, i, p)\) : \(i\)-ste Eigenschaft Cluster \(p\) in Cluster \(c\). Durch die Kombination der Sub-Formen der Lambdaformen erzeugt diese Funktion.
3 Entwurf


3.1 Ausgabedatei für USP


Abbildung 25: Ein Beispiel für eine MLN Datei
Abbildung 26: Ein Beispiel für eine PARSE Datei


3.2 Der erste Ansatz: Generierung der Regeln für Informationsextraktion


- Die sehr oft vorkommenden, gefundenen Wörter sind Produkteigenschaft, weil verschiedene Produkte sehr möglich die gleiche Produkteigenschaft haben.
- Die Produkteigenschaft bindet direkt mit Produkt und Produkteigenschaftswert bindet direkt mit Produkteigenschaft
- Produkteigenschaft sind am meisten die Nomen
- Produkte sind Nomen
- Verb ist kein Produkt, keine Produkteigenschaft, kein Produkteigenschaftswert
Die Regeln werden zwar schon gefunden, aber man muss die folgende Probleme lösen:

- Nach der Annahme, die sehr oft vorkommenden, gefundenen Wörter sind Produkteigenschaft, s. g. „oft“ muss möglichst gut definiert werden.
- Wie bindet Produkteigenschaft mit Produkt? Wie bindet Produkteigenschaft mit Produkteigenschaftswert? D.h. wie ist die Hierarchie der Wörter?
- Die Regeln, die mit der Hilfe von WEKA generiert werden, gelten theoretisch nur für die Daten, die in WEKA eingesetzt werden.


- Produkteigenschaften sind am meisten Nomen.
- Produkt ist Norm.
- Wenn ein Wort Worttyp „CD“ hat, ist das Wort sehr möglich ein Produkteigenschaftswert.
- Wenn ein Wort Worttyp „V“ hat, ist das Wort kein Produkt, keine Produkteigenschaft oder kein Produkteigenschaftswert.
- Produkt und Produkteigenschaft sind in Verbindung mit Argumentform „amod“
- Produkt und Produkteigenschaft sind in Verbindung mit Argumentform „appos“
- Produkt und Produkteigenschaft sind in Verbindung mit Argumentform „num“
- Produkt und Produkteigenschaft sind in Verbindung mit Argumentform „nn“
- Produkt und Produkteigenschaft sind in Verbindung mit Einheiten


Zwar werden die Regeln generiert, aber das Problem für die Unterscheidung zwischen Produkt und Produkteigenschaft immer noch nicht gelöst. Das Problem geht darauf wieder zurück, eine Hierarchie aufzubauen.
3.3 Der zweite Ansatz: Generierung der Hierarchie der Wörter


In Abbildung 11 wird die Ursache von Krebs beschreibt: wenn man raucht, leidet man an Krebs, d.h. Rauchen führt zu Krebs. Die formale Darstellung für die Aussage „Rauchen führt zu Krebs“ ist \( \forall x Sm(x) \Rightarrow Ca(x) \). Durch die Lambda Reduktion wird die Form gebildet: \( Sm(A) \Rightarrow Ca(A) \). Die graphischen Darstellungen sind:

![Graphische Darstellung von Markov Logik Netzwerk](image)

Abbildung 28: links ist die graphische Darstellung von Markov Logik Netzwerk. Rechts ist die gerichtete graphische Darstellung, entspricht der graphische Darstellung von MLN.


![Graphische Darstellung von "*.mln" Datei](image)

Abbildung 29: Ein weiteres Beispiel für graphische Darstellung von "*.mln" Datei

Um die Idee zu erklären, werde ich die Definitionen und Annahme wiederholen:

- Ein Blatt ist ein Knoten, der keine Kindknoten hat.
- Ein Wurzel Knoten ist der Knoten, der keine Elternknoten hat.
- Die Blätter sind die Produkteigenschaftswerte.
- Die Produkteigenschaftswerte verbinden sich nicht mit Verb Wörtern.
- Die Elternknoten der Blätter sind Produkteigenschaften.
- Die Wurzel ist entweder ein Produkt oder ein Markenzeichen.

Die Idee ist:

- Bestimmen der Abhängigkeit zwischen den Blätter und ihren Eltern Knoten.
- Nach der Annahme werden die Blätter in die Menge vom Produkteigenschaftswert hinzugefügt, und die direkten Elternknoten der Blätter werden in die Menge von Produkteigenschaft hinzugefügt.
- Bestimmen der Abhängigkeit zwischen den Knoten.
- Während der Bestimmung der Abhängigkeiten wird die Hierarchie gleichzeitig erstellt.
- Nach der Annahme werden die Wurzel Knoten in die Menge von Produkt hinzugefügt.
- Nach der Annahme werden alle nicht Normen Wort werden gefiltert.

Im Vergleich zu den vorherigen Ansätzen, z.B. [8], [10], [11], [12] und [13], Die Extraktion von Produkten, Produkteigenschaften und Produkteigenschaftswerten ist voll automatisch:

- Nur die Software für die Sprachanalyse ist erforderlich.
- Die Daten der Eingabe muss nicht mehr manuell markiert werden.
- Die Ontologie bzw. die Hierarchie der Wörter wird bei Analyse vom Netz der Wörter erstellt.
- Dieses Text Mining Verfahren gilt für natürlichen Text und offene Domänen.
4 Implementierung

4.1 Generierung der Hierarchie durch Bestimmung der Abhängigkeit von den Wörtern

Die Hauptaufgabe dieses Ansatzes ist die Bestimmung der Abhängigkeit zwischen dem Knoten, damit eine Hierarchie der Wörter erstellt wird. Danach wird die Hierarchie verbessert, schließlich werden die bestimmten Knoten in dem Graph gefiltert.

Einige Definitionen:
- Wenn ein Knoten Count und ClusterSum hat, kommt dieser Knoten aus dem Bereich „Argumente von Core Form“ (siehe in der Abbildung 30). „apple“, „pro“ und „display“ haben Count und ClusterSum. „announce“ hat ClusterSum aber kein „Count“.
- Cluster ID von Knoten aus dem Bereich „Argumente von Core Form“ wird CClusterID genannt, ClusterSum von Knoten aus dem Bereich „Argumente von Core Form“ wird CClusterSum genannt.
- CC = Count/CClusterSum

Abbildung 30: Illustration für einige Definitionen

4.2 Vorverarbeitung

In diesem Schritt wird die Abhängigkeit zwischen den Blättern und Elternknoten bestimmt. Wenn die Hierarchie zwischen den Blättern und Elternknoten festgelegt wird, können die Produkteigenschaften und die Produkteigenschaftswerte nach der Annahmen in „property“ Menge hinzugefügt werden. Die Werte von CC werden für alle Knoten berechnet. Um die Hierarchie in dem nächsten Schritt weiter zu erstellen, müssen die Werte von Count und CClusterSum von Kindknoten und ClusterSum von Elternknoten aktualisiert werden.

4.2.1 Verschiedene Fälle

Um die Abhängigkeit zwischen den Knoten zu bestimmen, werden die folgende Situationen betrachtet:
• Viele Knoten kommen nur einmal im Graph vor. Solche Knoten kommen im Graph vor, genau dann wenn ihre Nachbarknoten im Graph vorkommen. Es gibt eine Ursachlichkeit bzw. Abhängigkeit dazwischen.

• Manche Knoten kommen einige Male im Graph vor. Die Knoten kommen im Graph vor, genau dann wenn ihre Nachbarknoten im Graphen vorkommen. Es gibt eine Ursachlichkeit bzw. Abhängigkeit dazwischen.

• Manche Knoten kommen einige Male im Graph vor. Der Knoten kommt im Graph am meisten zusammen mit ihrem Nachbarknoten vor. Es gibt eine Ursachlichkeit bzw. Abhängigkeit dazwischen.

Abbildung 31 illustriert die o.g. Fälle:
• „1920x1080“ kommt vor, genau dann wenn „display“ vorkommt.
• „retina“ kommt vor, genau dann wenn „display“ vorkommt.
• wenn „ips“ vorkommt, kommt „display“ am meisten vor.

Abbildung 31 : Ein Teil von “*.mln“

4.2.2 Bestimmung der Abhängigkeit

CClusterSum und Count von „1920x1080“ sind 1. Das bedeutet:
• „1920x1080“ kommt nur einmal in diesem Text vor. „1920x1080“ kommt einmal als Nachbarknoten von „display“ vor. Das Vorkommen von „1920x1080“ ist nur abhängig von dem Vorkommen von „display“.

CClusterSum und Count von „retina“ sind 2. Das bedeutet:
• „retina“ kommt zweimal in diesem Text vor. „retina“ kommt zweimal als Nachbarknoten von „display“ vor. Das Vorkommen von „retina“ ist nur abhängig von dem Vorkommen von „display“.

Count von „ips“ ist 2 und CClusterSum von „ips“ ist 3. Das bedeutet:
• „ips“ kommt sehr oft zusammen mit dem Nachbarknoten „display“ vor. (Das Problem ist hier, wie man „oft“ möglichst gut definieren kann. Das ist eine Arbeit für die Zukunft. Hier wird die „oft“ als „Count/ClusterSum >=0.5“ definiert)

Verallgemeinerte Bedeutung:
\[ C_{ClusterSum} = Count \Rightarrow \text{Es gibt eine starke Abhängigkeit zwischen den Knoten. Wenn die Knoten aus dem Bereich „Argumente von Core Form“ im Text vorkommen, genau dann wenn ihre Nachbarknoten im Text vorkommen.} \]

\[ CC >= 0.5 \Rightarrow \text{Es gibt eine leichte Abhängigkeit zwischen den Knoten, die Abhängigkeit ist noch nicht deutlich.} \]

Nach o.g. Aussage wird die Hierarchie der Knoten bestimmt. Der Knoten mit \( CC=1 \), ist ein Kindknoten von seinem Nachbarknoten. Abbildung 32 illustriert die Erstellung einer Hierarchie durch die Bestimmung der Abhängigkeit zwischen den Knoten. Nach der Bestimmung der Abhängigkeiten entstehen die Blätter in dem Graph. Der rote Knoten ist der Kindknoten und die Knoten „display“ ist Elternknoten von Kindknoten.

Abbildung 32 : Oben ist die graphische Darstellung von Abbildung 31. Nach der Bestimmung der Abhängigkeiten (\( CC=1 \)) wird die graphische Darstellung (unten) erzeugt.


Abbildung 33 : Es gibt die Abhängigkeit zwischen Knoten „ips“ und „display“, aber die Abhängigkeit ist nicht deutlich

37
4.2.3 Bestimmung von Produkteigenschaften und Produkteigenschaftswerten

- Nach der Annahme sind alle Knoten, die Worttyp „V“ besitzt, keine Produkte, keine Produkteigenschaften und keine Produkteigenschaftswerte.
- Nach der Annahme sind die Blätter die Produkteigenschaftswerte.
- Nach der Annahme sind die Elternknoten der Blätter Produkteigenschaften.
- Nach der Annahme allen Knoten, die mit Verb Wörtern sich verbinden, sind keine Produkteigenschaftswerte.
- Wenn ein Cluster keine Kindcluster besitzt, z.B. „31“ in der Abbildung 34, sind die entsprechenden Cluster Knoten die Blätter.

Abbildung 34 : Beispiel für die Darstellung von Blättern in „*.mln“, 31 ist ein Blatt und 30 ist kein Blatt.

Die gefundenen Knoten werden durch o.g. Annahmen gefiltert, schließlich werden die übrigen Knoten in einer „property“ Menge hinzugefügt. Um die Produkteigenschaften und Produkteigenschaftswerte in einer Menge zu unterscheiden, wird jedem Knoten ein Wert zugewiesen, Produkteigenschaft entspricht den Wert 0 und Produkteigenschaftswert entspricht Wert -10.

4.2.4 Aktualisierung der Werte von ClusterSum, Count und CClusterSum

Wenn ein Knoten als Kindknoten oder Elternknoten benannt ist, bedeutet dies, die Abhängigkeit zwischen den Knoten und seinem Nachbarknoten bereit bestimmt ist.

- Der Knoten hat nur ein Blatt ⇒ ClusterSum = ClusterSum – Count, CClusterSum=0 und Count=0.
- Der Knoten besitzt mehrere Knoten ⇒
- `ClusterSum=Clustersum - |processednode|` (Größe der Menge „processednode“).
- `Count` von jedem Kindknoten wird 0, d.h. `Count=0`.
- `CClusterSum` von jedem Kindknoten wird 0, d.h. `CClusterSum=0`.

Sei WordID Menge vom Wort „display“ `{000:0:003, 000:0:050}`. Seien `searchNode(retina, display)=\{000:0:050\}`, `searchNode(ips, display)=\{000:0:003, 000:0:050\}` und `searchNode(1920*1080, display)=\{000:0:050\}`, daraus folgt `processednode=\{000:0:003, 000:0:050\}` und `|processednode|=2`.

<table>
<thead>
<tr>
<th></th>
<th>ClusterSum</th>
<th>CClusterSum</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>display</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>retina</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1920*1080</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ips</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Abbildung 35: Ein Beispiel für die Aktualisierung der Werte von `ClusterSum`, `Count` und `CClusterSum`.

<table>
<thead>
<tr>
<th></th>
<th>ClusterSum</th>
<th>CClusterSum</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>display</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>retina</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1920*1080</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ips</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>


In diesem Schritt werden alle Abhängigkeiten von den Knoten mit `CC=1` bestimmt, und die Produkteigenschaften und Produkteigenschaftswerte gefunden. Die Knoten mit `CC>=0.5` werden auch ausgewählt, es gibt die Abhängigkeiten zwischen diesen Knoten und ihren Nachbarknoten. Um die nicht deutlichen Abhängigkeiten zu bestimmen, müssen die Werte von `ClusterSum`, `Count` und `CClusterSum` aktualisiert werden.
4.3 Bearbeitung der Knoten mit CC>=0.5

Nach der Aktualisierung der Werte von ClusterSum, Count und CClusterSum entstehen 4 Fälle, durch die Analyse von diesen 4 Fällen werden die Abhängigkeiten bestimmt.

4.3.1 Vier Fälle

4.3.1.1 Der erste Fall

CClusterSum=0 und Count=0

Bedingung (1)

Sei initClusterSum ein niemals aktualisierte ClusterSum, d.h. der Wert kommt aus „*.mln“, initCClusterSum sei ein niemals aktualisierte CClusterSum. d.h. der Wert kommt aus „*.mln“. Die Abhängigkeit wird dadurch bestimmt:

- Solcher Fall tritt auf, wenn ein Knoten zu mehreren Nachbarknoten gehört. Der Knoten, der die Bedingung (1) erfüllt, ist ein Kindknoten von ihrem Nachbarknoten, d.h. Nachbarknoten ist Elternknoten, z.B. „2560*1600“ in der Abbildung 37 und 38.
4.3.1.2 Der zweite Fall

*CClusterSum*>0 und *Count*=0  

Bedingung (2)

Wenn ein Knoten, der die Bedingung (2) erfüllt, im Text vorkommt, kommt ihr Nachbarknoten auch vor.

Abbildung 39: "*.mln" für 2.Fall


Sei \textit{initClusterSum} eine niemals aktualisierte \textit{ClusterSum}, \textit{intCClusterSum} eine niemals aktualisierte \textit{CClusterSum}. Die Abhängigkeit wird dadurch bestimmt:

- Wenn ein Knoten die Bedingung (2) erfüllt und \textit{initClusterSum} >= \textit{intCClusterSum}, dann ist dieser Knoten ein Kindknoten von Nachbarknoten.
- Wenn ein Knoten Bedingung (2) erfüllt und \textit{initClusterSum} < \textit{intCClusterSum}, dann ist dieser Knoten ein Elternknoten von Nachbarknoten.

Abbildung 40: Ein Beispiel für 2.Fall

4.3.1.3 Der dritte Fall

\textit{CClusterSum}=0 und \textit{Count}>0 Bedingung (3)

Wenn ein Knoten, der die Bedingung (3) erfüllt, im Text vorkommt, kommt ihr Nachbarknoten auch vor.

Abbildung 41: "*.mln" für 3.Fall
Der vierte Fall

CClusterSum > 0 und Count > 0

Wenn ein Knoten A nur einen Nachbarknoten B, der die Bedingung (4) erfüllt, hat, dann ist Knoten A ein Kindknoten von Knoten B und B ist ein Elternknoten. Wenn ein Knoten mehrere Nachbarknoten hat, welche die Bedingung (4) erfüllen können, wird dieser Knoten im nächsten Schritt weiter verarbeitet.

Abbildung 42: Ein Beispiel für 3. Fall

Abbildung 43: *.mln" für 4. Fall

Abbildung 44: Ein Beispiel für die Bestimmung der Abhängigkeiten. Links ist die Graphische Darstellung für ein Cluster, rechts ist eine Tabelle für ClusterSum, CClusterSum und Count.
In der Abbildung 44, die Abhängigkeiten zwischen „take“ und „except“ sowie „wrap“ werden in Vorverarbeitungsschritt bestimmt, die Abhängigkeiten zwischen „take“ und „pro“ wird nach der Bedingung (3) bestimmt, und die Abhängigkeiten zwischen „take“ und „apple“ wird nach der Bedingung (4) bestimmt.

4.3.2 Aktualisierung der Werte von ClusterSum, Count und CClusterSum

Der Verfahren der Aktualisierung der Werte von \textit{ClusterSum}, \textit{Count} und \textit{CClusterSum} ist gleich wie der Fall „Der Knoten besitzt mehrere Knoten“ in 4.2.4.

4.4 Verarbeitung der Konten mit CC<0.5

4.4.1 Bestimmung der Abhängigkeit

Bis jetzt kann die Abhängigkeit zwischen den Knoten und ihren Nachbarknoten mit \textit{CC}>=0.5 bestimmt werden. Es gibt einige Knoten mit \textit{CC}<0.5. Die Idee für die Verarbeitung von dem Knoten mit \textit{CC}<0.5 ist : man aktualisiert die Werte von \textit{Count}, \textit{CClusterSum} und \textit{ClusterSum} bis zu \textit{CC} Wert>=0.5, damit die vorherige Ansätze wieder verwendbar sind.

Abbildung 45 : ein Beispiel für die Bestimmung der Abhängigkeiten. Links ist die Graphische Darstellung für ein Cluster, rechts ist eine Tabelle für \textit{ClusterSum}, \textit{ClusterSum} und \textit{Count}. 

\begin{center}
\begin{tabular}{|l|c|c|}
\hline
ClusterSum & CClusterSum & Count \\
\hline
pro & 5 & 1 \\
\hline
generation & 2 & 1 \\
\hline
13 & 4 & 3 \\
\hline
macbook & 6 & 5 \\
\hline
retina & 6 & 1 \\
\hline
previous & 2 & 1 \\
\hline
new & 4 & 1 \\
\hline
they & 2 & 1 \\
\hline
display & 5 & 1 \\
\hline
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{|l|c|c|}
\hline
ClusterSum & CClusterSum & Count \\
\hline
pro & 0 & 0 \\
\hline
generation & 0 & 0 \\
\hline
13 & 1 & 0 \\
\hline
macbook & 1 & 0 \\
\hline
retina & 4 & 1 \\
\hline
previous & 0 & 0 \\
\hline
new & 2 & 1 \\
\hline
they & 0 & 0 \\
\hline
display & 2 & 1 \\
\hline
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{|l|c|c|}
\hline
ClusterSum & CClusterSum & Count \\
\hline
pro & 0 & 0 \\
\hline
generation & 0 & 0 \\
\hline
13 & 0 & 0 \\
\hline
macbook & 0 & 0 \\
\hline
retina & 2 & 0 \\
\hline
previous & 0 & 0 \\
\hline
new & 0 & 0 \\
\hline
they & 0 & 0 \\
\hline
display & 0 & 0 \\
\hline
\end{tabular}
\end{center}
„display“, „new“ und „retina“ besitzt den Wert CC>0.5, durch maximal zwei Aktualisierung der Werten ClusterSum, ClusterSum und Count werden die CC Werte bis zu kleiner gleich 0.5 reduziert.

Schließlich gibt es keine Knoten mit CC<0.5. Wenn es keine Knoten mit CC<0.5 gibt, wird die Hierarchie der Knoten erstellt. Ein ungerichteter Graph in einen gerichteten Graph umgewandelt.

Das Verfahren von der Aktualisierung der Werte von ClusterSum, Count und CClusterSum ist gleich wie der Fall „Der Knoten besitzt mehrere Knoten“ in 4.2.4.

### 4.4.2 Aktualisierung der Werte von ClusterSum, Count und CClusterSum

Im Vorverarbeitungsschritt werden die CC Werte berechnet. Wenn der CC Wert größer gleich 0.5 ist, dann werden entsprechende Knoten in mnm(Merged Node Map) hinzugefügt, die aktualisierten Werte von ClusterSum werden in mergedmap_c2s hinzugefügt und die aktualisierter Werte von CClusterSum und Count werden in mergedmap_c2cnsc hinzugefügt.

Einige Definitionen:

- **mnm** : LinkedHashMap
  
  \[ \text{ClusterID} = \{\text{ClusterID, die entsprechende Knoten mit CC} \geq 0.5\} \]

- **mergedmap_c2s** : LinkedHashMap
  
  \[ \text{ClusterID} = \text{neuer ClusterSum} \]

- **mergedmap_c2cnsc** : LinkedHashMap
  
  \[ \text{ClusterID} = \{\text{ClusterID!neuer ClusterSum!neue Count}\} \]

Aktualisierung der Werte von ClusterSum, Count und CClusterSum für die Knoten mit CC≠1 werden dadurch erledigt, mnm, mergedmap_c2cnsc und mergedmap_c2cnsc zu aktualisieren.

- Geht von mergedmap_c2s aus. Wenn CC Wert größer gleich 0.5, dann wird entsprechende Knoten in mnm hinzugefügt. Bemerkung: Die Bedingung (n2sum != 0 && n2count != 0) implementiert 4. Fall in 4.2.2.1. Nur die Knoten, die 4. Fall erfüllt, werden verarbeitet, weil die Abhängigkeiten von anderen Knoten schon festgelegt werden.

```
while (mergedmap_c2cnsc is not empty)
{
  if (Count/CClusterSum>=0.5) {
    add ClusterID into mnm
  }
}
```

Abbildung 46: Pseudocode für Berechnung von CC Wert und mnm

- Geht von mergedmap_c2s aus, wenn der Knoten kein Blatt ist, wird die Menge von „processednode“ mithilfe der Funktion searchNode berechnet. Die Größe der Menge

45

Abbildung 47: Pseudocode für Berechnung von neuer ClusterSum


Abbildung 48: Pseudocode für Berechnung von neuem Count

4.5 Verbessern die Hierarchie

Zwar werden die Abhängigkeiten bis jetzt schon bestimmt, aber es gibt immer noch einige Fehler. Mit folgenden Funktionen werden die Fehler korrigiert.

Abbildung 49: Beispiel für eine falsche Bestimmung der Abhängigkeit zwischen den Knoten

Knoten A und Knoten B „nn“ ist, wird die Anhängigkeit zwischen Verb und Knoten B gelöscht.

![Diagramm Knoten Beziehung](image)

**Abbildung 50 : Beispiel für eine Korrektur der Abhängigkeit**

- Wenn die Knoten die in Abbildung 51 dargestellte Beziehung haben und Knoten A hat keine Beziehungen mit anderen Knoten, wird die Abhängigkeit zwischen den Knoten korrigiert. Diese Abbildung 51 illustriert einen solchen Fall.

![Diagramm Knoten Beziehung](image)

**Abbildung 51 : Beispiel für Korrektur der Abhängigkeit**


Abbildung 52: Beispiel für Korrektur der Abhängigkeit


Abbildung 54: Beispiel für die Entfernung eines Zyklus

Durch o.g. Verfahren wird die Hierarchie korrigiert. Um eine bessere Hierarchie zu erhalten, ist eine Erweiterung erforderlich. Einige Regeln, die in 3.3 generiert werden, kann man zum Korrektur der Abhängigkeit benutzen. Aus meiner Erfahrung ist die Korrektur der Abhängigkeit mit einzelnen Regeln, ist es möglich, dass die richtige Hierarchie falsch korrigiert wird.

4.6 Filter

Alle gefundene Produkt, Produkteigenschaft und Produkteigenschaftswerte werden noch einmal gefiltert, um die Klassifikation der Wörter zu verbessern.

- Einige Wörter, die als Produkteigenschaftswerte erkannt werden, sind keine Produkteigenschaftswerte. Diese Wörter in der Hierarchie sind Kindknoten von Produkteigenschaften, aber sie haben die Beziehung zwischen den diesen Kindknoten
und Produkteigenschaften, d.h. die Argumentform ist „nn“. Wenn die Wörter die Argumentform „nn“ haben, sind die Wörter eine Nominalphrase. Das heißt, wenn die Elternknoten eine Produkteigenschaft ist, ist die Kindknoten nicht ein Produkteigenschaftswert sondern eine Produkteigenschaft, weil die Nominalphrase aus diesen Wörtern besteht. Es geht von Eigenschaftswert in „property“ Menge aus, wenn das Wort des Eigenschaftswerts mit seinem Elternknoten die Argumentform „nn“ hat, dann wird das Wort des Eigenschaftswerts als Eigenschaft korrigiert. Abbildung 54 illustriert das Verfahren von Filter.

![Diagramm 1](image1.png)

Abbildung 55: Beispiel für den Filter


![Diagramm 2](image2.png)

Abbildung 56: Beispiel für den Filter

Einige Wörter, die als Produkte erkannt werden, sind keine Produkte. Diese Wörter in der Hierarchie sind Wurzel Knoten, haben aber eine Beziehung „conj_and“ oder „conj_or“ zwischen den Wörtern und ihre Kindknoten (d.h. die Argumentform ist „conj_and“ oder „conj_or“). Wenn ihre Kindknoten Produkteigenschaften sind, ist es möglich, dass die Wurzel Knoten ebenfalls Produkteigenschaft sind. Wenn ein Kindknoten einer Wurzel, ausgehend von einer Wurzel Menge aus, eine Produkteigenschaft ist, dann wird diese Wurzel als Produkteigenschaft bezeichnet. Abbildung 57 illustriert das Verfahren des Filterns.

Der Filter hat keinen Einfluss auf die Änderung der Hierarchie, deshalb hat die Korrektheit der Hierarchie einen großen Einfluss auf die Filter. Wenn das Produkt, Produkteigenschaft und Produkteigenschaftswert nicht richtig erkannt werden, werden die Ergebnisse von Filter auch nicht exakt.

### 4.7 Ausgaben Filter

Einige Wurzel Knoten keine Normen sind. Nach der Annahme in 3.3 werden alle Wörter, die keine Normen sind, gefiltert.
### 5 Experiment


![Diagram](image)

Abbildung 59: Eine Darstellung für ein Produkt, sei der Produktname „BAC“, „A“ repräsentiert eine Kategorie

Die Genauigkeit von Produkt, Produkteigenschaft und Produkteigenschaftswert werden in der folgenden Abbildungen gegeben. Die Genauigkeit beschreibt den Anteil von relevanten Wörtern an der Menge von gefundenen Wörtern:

\[
P = \frac{\text{relevante Wörter}}{\text{gefundene Wörter}}
\]


6 Zusammenfassung

7 Ausblick

Durch Erstellung einer Hierarchie der Wörter werden die Produkte, Produkteigenschaften und Produkteigenschaftswerte extrahiert, zwar ist die Genauigkeit nicht gut, aber es ist möglich zu erweitern.

7.1 Nicht löschrabe Probleme


Abbildung 65: Links ist die falsche Darstellung, rechts ist die richtige Darstellung.

- Wenn die Argumentform zwischen Kindknoten und Elternknoten „nn“ ist, dann ist die Abhängigkeit zwischen den Knoten schwierig zu bestimmen.

Abbildung 66: Beispiel für zwei Wörter, die Argumentform „nn“ haben.

7.2 löscharbare Probleme

Wie in „4.5 Verbessern die Hierarchie“ gezeigt, gibt es einige löscharbare Probleme.

- Wenn zwei Wurzeln oder zwei Elternknoten die gleichen Produkteigenschaften haben, sind die zwei Knoten sehr wahrscheinlich ein gleiches Produkt, z.B. „macbook“ und „device“.

Um die Produkteigenschaft und den Produkteigenschaftswert zu unterscheiden, kann man die Regel für Informationsextraktion, die in 3.2 generiert wird, man benutzen. Ein Voting Algorithm(mehr siehe 26) kann man benutzen, damit die Regeln bzw. die Randbedingungen entspannt werden.


Abbildung 67: Die Verbesserung der Hierarchie mit Argumentform „conj“.

7.3 Arbeit für die Zukunft

In dieser Diplomarbeit wird die Bestimmung der Abhängigkeit eine Abfolge der Bedingungen benutzt. Das heißt, wenn eine Bedingung verletzt, wird die Abhängigkeit falsch bestimmt. Besser kann man den Voting Algorithmus benutzen, um die Hierarchie besser aufzubauen. Es

7.4 Begrenzung der System-Anforderung

8 Literaturverzeichnis


61
Anhang A: Der Ergebnis von WEKA: (Aus diesem Ergebnis werden die Regeln (in 3.2) für Informationsextraktion generiert.)
Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben.
Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken
übernommene Aussagen als solche gekennzeichnet.
Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens.
Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht.
Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Unterschrift:

Stuttgart, 06.05.2013