

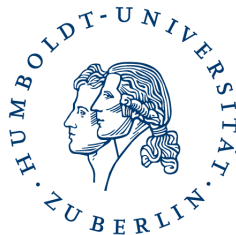
- *Studienarbeit* -

Akronymgenerierung mit statistischen Modellen

Betreuer: Prof. Tobias Scheffer, Steffen Bickel

Eingereicht von:
Marco Oppel
6. April 2006

Humboldt-Universität zu Berlin
Institut für Informatik
Lehrstuhl für Wissensmanagement



Zusammenfassung

In dieser Studienarbeit untersuche ich, inwieweit Verfahren aus der maschinellen Übersetzung auf Grundlage des Noisy-Channel-Modells geeignet sind, Akronyme zu generieren. Ich unterscheide dabei zwischen Realwort- und Scheinwort-Akronymen, wobei im ersten Fall das Akronym bzw. die Abkürzung selbst wieder ein echtes Wort ist und im zweiten Fall ein Kunstwort, das nicht in einem Wörterbuch steht. Für die Generierung von Realwort-Akronymen nutze ich naheliegender Weise ein Wörterbuch, was auch relativ gut funktioniert. Ein flexibleres N-Gram-Modell kam für die Aufgabe der Generierung von Scheinwort-Akronymen zum Einsatz, hat sich allerdings als wesentlich fehleranfälliger.

Inhalt

1. Einleitung	4
2. Problemstellung	4
3. Vorbetrachtung	4
3.1. Noisy-Channel-Modell.....	5
3.2. Maschinelle Übersetzung.....	7
3.3. N-Gram-Modelle.....	7
4. Modell für Akronymgenerierung	8
4.1. Was ist ein Akronym?.....	8
4.2. Modell.....	10
4.3. Akronymmodell.....	10
4.3.1. Wörterbuchmodell.....	10
4.3.2. N-Gram-Modell.....	10
4.4. Buchstabenzuordnungsmodell.....	12
4.5. Semantische Klassen.....	13
5. Dekodierung	14
5.1. Suche nach echten Wörtern.....	15
5.2. Suche nach Scheinwörtern.....	16
6. Experimentelle Untersuchung	18
6.1. Akronymgenerierung echter Wörter.....	19
6.2. Akronymgenerierung von Scheinwörtern.....	21
7. Auswertung	22
Anhang A: Ergebnisse Generierung Scheinwort-Akronyme	25
Anhang B: Verzeichnisse	34

1. Einleitung

„Ein aus den Anfangsbuchstaben mehrerer Wörter gebildetes Wort“, so beschreibt der Duden den Begriff Akronym, aussprechbare Abkürzungen von meist sehr langen und komplizierten Bezeichnungen. Dabei liegt die besondere Eleganz in der Balance zwischen Wortcharakter und Abkürzung der Ursprungswörter, Akronyme lassen sich oft so verblüffend einfach aussprechen wie ein normales Wort. Ein wohl bekanntes Beispiel für ein Akronym ist *NATO*, das wie geschrieben ausgesprochen wird und für „North Atlantic Treaty Organisation“ steht.

Einige Akronyme wie *Radar*, *Aids* oder *Laser* sind so gebräuchlich, dass sie es selbst in das große deutsche Wörterbuch geschafft haben. Außerdem können Akronyme besonders leicht einprägsam sein und eine hohe Originalität aufweisen, wie es bei Markennamen wie etwa *Haribo*, *Ikea*, *Degussa* oder *Lego* der Fall ist.

Doch wie findet man solch gut aussprechbare, treffende und originelle Akronyme? In dieser Arbeit untersuche ich, inwieweit statistische Verfahren geeignet sind, zu vorgegebenen Wörtern Akronyme zu finden.

2. Problemstellung

Die Aufgabe lautet folgendermaßen: Finde zu einer vorgegebenen Wortgruppe das beste Akronym oder die n besten Akronyme. Dabei bestimmen zwei wesentliche Charakteristika die Güte eines Akronyms, zum Einen die Aussprechbarkeit und zum Anderen die Eignung als Abkürzung der Ursprungswörter.

Eine ähnliche Problemstellung findet sich bei der maschinellen Übersetzung, also bei den Verfahren der elektronischen Übersetzung von Texten aus einer Sprache in eine andere. Für einen übersetzten Satz soll dabei einerseits gelten, dass eine hohe Übereinstimmung mit der ursprünglichen Bedeutung herrscht und andererseits die Übersetzung in der Zielsprache grammatikalisch korrekt ist. Diese Unterscheidung ist wichtig, weil sich syntaktische Regeln wie der Satzbau zwischen den Sprachen unterscheiden können. Bei den Akronymen entspricht diese Unterscheidung in etwa der Eignung als Abkürzung und der Aussprechbarkeit, deshalb scheinen mir die Verfahren aus der maschinellen Übersetzung geeignet zu sein, für die Akronymgenerierung angewandt zu werden.

Ich unterscheide zwischen den beiden Gruppen der Realwort- und Scheinwortakronyme, wobei erstere selbst echte Wörter sind, die in einem Wörterbuch quasi nachgeschlagen werden können. Im zweiten Fall ist ein geeignetes abstrakteres Modell für das Akronym einzubeziehen.

Die Güte der Generierungsmodelle wird anhand der Anzahl der erfolgreich generierten Akronymen einer Testmenge gemessen.

3. Vorbetrachtung

Es finden sich eine Menge wissenschaftlicher Untersuchungen zu dem Thema, Akronyme und Abkürzungen aus Texten zu erkennen und zu extrahieren. Hieran besteht ein großer Bedarf, da die Anzahl wissenschaftlicher Veröffentlichungen im biomedizinischen Bereich sehr groß ist und eine unüberschaubare Vielzahl auch mehrdeutiger Abkürzungen darin benutzt wird. So wurden laut [Medline05] allein im Jahr 2004 über 571.000 neue Einträge in das System Medline¹ der nationalen medizinischen Bibliothek der USA aufgenommen. An fünf Tagen in der Woche kommen jeweils 1500 bis 3500 neue Verweise hinzu und aufgrund der Datenmengen werden in den letzten beiden Monaten im Jahr keine neuen Einträge vorgenommen, sondern für das Aktualisieren von Index und Thesaurus genutzt.

¹ Medline (Medical Literature Analysis and Retrieval System Online) ist ein online System zur Literaturanalyse und -suche und ein wichtiger Bestandteil der PubMed Datenbank (<http://pubmed.gov/>).

Somit ist der Wunsch nach einer Datenbank mit Abkürzungen und Akronymen verständlich, um mit den vielen Veröffentlichungen einigermaßen Schritt halten zu können. Eine solche Datenbank ist beispielsweise AcroMed, deren Konstruktion in [PUST02] beschrieben wird. Diese Datenbank wurde aus den Zusammenfassungen der Artikel der Medline Datenbank konstruiert, in denen Paare von Abkürzungen und ausgeschriebenen Formen gesucht und gespeichert wurden. Abkürzungen in biomedizinischen Veröffentlichungen sind dabei recht schwierige Gebilde, da sie nicht nur über Buchstaben gebildet werden wie bei *NATO*, sondern auch über Zahlen und Symbole beispielsweise CaSR für *Ca²⁺-sensing receptor* oder E2 für *estradiol-17 beta*. Die Gruppe um Pustejovsky benutzt ein Verfahren das auf Mustern und regulären Ausdrücken beruht, um potentielle Abkürzungen und deren Bedeutung in Texten zu finden. Verbessern konnten sie dieses Verfahren durch die Einbeziehung syntaktischer Informationen. Also indem sie Text verwenden, der bereits syntaktisch annotiert und in bedeutungstragende Gruppen aufgeteilt ist.

Ein weiteres sehr einfaches und schnelles Verfahren, um Abkürzungen und deren ausgeschriebene Form in Texten zu finden, wird in [SCH02] vorgestellt. In einem ersten Schritt wird hier versucht, die Paare bzw. Kandidaten dafür zu finden, in dem nach dem Muster *lange Form (Abkürzung)* oder *Abkürzung (lange Form)* gesucht wird. Es wird also nach Wörtern in Klammern gesucht und links davon nach der langen Form oder der Abkürzung. Der zweite Schritt sucht zu einer Abkürzung die beste ausgeschriebene Form, indem zu den Buchstaben in der Abkürzung von rechts nach links die entsprechenden Buchstaben in den ausgeschriebenen Wörtern gesucht werden. Dabei gibt es nur die Bedingung, dass die Buchstaben übereinstimmen müssen, die Zuordnung von Buchstaben kann also auch mitten im Wort liegen. Eine weitere Bedingung betrifft den ersten Buchstaben der Abkürzung, der mit einem Wortanfang zusammenfallen soll.

Ein Verfahren zur Generierung von Akronymen wird in [TSU05] vorgestellt. Dort wird die Generierung von Akronymen als Markierungsaufgabe aufgefasst. Die abzukürzenden Wörter werden buchstabenweise mit den Markern SKIP (Buchstabe kommt nicht im Akronym vor), UPPER (Buchstabe kommt als Großbuchstabe im Akronym vor), LOWER (Buchstabe kommt als Kleinbuchstabe im Akronym vor), SPACE (Buchstabe wird als Leerzeichen im Akronym interpretiert) und HYPHEN (aus dem Buchstaben wird im Akronym ein Bindestrich) markiert. Dabei werden alle Zeichen der Quellwortfolge markiert, also auch Zahlen und Leerzeichen. Aus den Markierungen wird dann das Akronym zusammengesetzt. Das Modell für die Markierung an sich basiert auf einem Maximum Entropie Markov Modell, das für die Entscheidung über die Art der Markierung eines Buchstabens viele Attribute betrachtet. Diese reichen von Buchstaben-N-Grammen über die letzte Markierung und der Groß- oder Kleinschreibung des Buchstabens bis hin zur Anzahl der abzukürzenden Worte, der Buchstabenfolge innerhalb des Wortes und der Entfernung des Buchstabens zu Wortanfang und -ende.

All diese Verfahren sind darauf ausgerichtet, Abkürzungen aus dem biomedizinischen Bereich zu verarbeiten, die nicht aussprechbar sein müssen. In dieser Arbeit geht es jedoch um aussprechbare Akronyme aus dem allgemeineren Sprachgebrauch. Das heißt, dass die Akronyme, die ich generieren möchte, aussprechbar sein und nur aus Buchstaben bestehen sollen, nicht etwa Zahlen enthalten so wie in den obigen Beispielen.

Im Folgenden möchte ich zunächst das Noisy-Channel Modell und seine Anwendung in der maschinellen Übersetzung vorstellen und anschließend meine Ansätze für die Akronymgenerierung erläutern.

3.1. Noisy-Channel-Modell

Das Noisy-Channel-Modell stammt aus dem Gebiet der Informationstheorie und modelliert die Übertragung von Informationen über einen rauschenden Kanal. Wie in Abbildung 1 dargestellt, wird dabei eine Nachricht n über einen Encoder in ein Signal s transformiert, dieses über einen Kanal gesendet und anschließend auf der Empfängerseite als Signal s' zu Nachricht m decodiert.

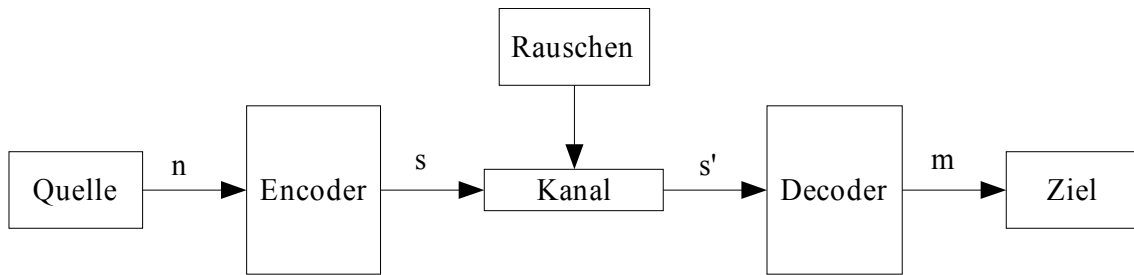


Abbildung 1: Noisy-Channel-Modell in der Informationstheorie

Während der Übertragung ist die Wahrscheinlichkeit p dafür, dass ein Bit invertiert und die Nachricht verfälscht wird. Dabei kann die Wahrscheinlichkeit, trotz des systembedingten Rauschens eine Nachricht mit geringer Fehlerwahrscheinlichkeit zu übertragen, erhöht werden, indem der Encoder geschickt redundante Informationen zu Nachricht n hinzufügt. Eine sehr einfache Möglichkeit ist etwa, jedes Bit dreimal zu senden und den Decoder eine Mehrheitsentscheidung durchführen zu lassen. Andererseits dauert dann die Übertragung von n auch dreimal so lang, der Anteil der Nutzdaten sinkt also mit zunehmenden Korrekturdaten.

Mit dem Zusammenhang von möglicher Übertragungskapazität und erreichbarer Fehlerrate beschäftigt sich das Noisy-Channel-Kodierungs-Theorem.

Quelle: [MAC03]

In der maschinellen Sprachverarbeitung kommt das Noisy-Channel-Modell auch zum Einsatz, wobei vor allem der Aspekt der Dekodierung aufgegriffen wird. Im Kontext von maschineller Übersetzung wird etwa angenommen, das Rauschen des Kanals hätte die Nachricht übersetzt und die Aufgabe des maschinellen Übersetzers wäre nun, aus der übersetzten Nachricht die Nachricht in der Ursprungssprache herauszufinden. Diese Herangehensweise, die in Abbildung 2 skizziert ist, kann in ähnlicher Form auf viele Aufgabenfelder übertragen werden.

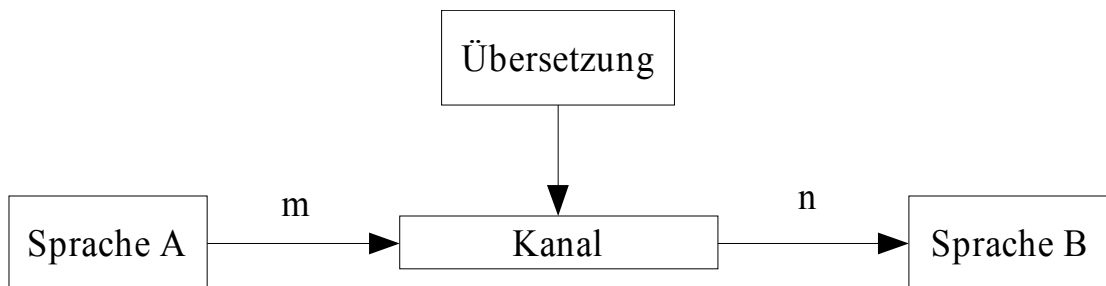


Abbildung 2: Noisy-Channel-Modell in der maschinellen Übersetzung

Das Rekonstruieren der ursprünglichen Nachricht lässt sich in folgender Form als Optimierungsaufgabe darstellen:

$$m = \underset{m}{\operatorname{argmax}} P(m|n) \quad (1)$$

Gesucht ist also diejenige Nachricht m , welche gegeben die empfangene Nachricht n am wahrscheinlichsten ist. Nach Bayes' Theorem lässt sich die bedingte Wahrscheinlichkeit $P(m|n)$ umformen zu $P(m|n) = \frac{P(n|m) \cdot P(m)}{P(n)}$. In (1) eingesetzt ergibt sich:

$$m = \underset{m}{\operatorname{argmax}} \frac{P(n|m) \cdot P(m)}{P(n)} = \underset{m}{\operatorname{argmax}} P(n|m) \cdot P(m) \quad (2)$$

Der Divisor $P(n)$ kann entfallen, da diejenige Nachricht m gesucht wird, welches die maximale Wahrscheinlichkeit liefert und $P(n)$ nicht von m abhängt.

Die Wahrscheinlichkeiten müssen dabei für jedes Problem passend definiert werden; sie modellieren eine gewisse Art Vorwissen welches über den jeweiligen Problemkontext bereits bekannt ist.

Quelle: [MAN03]

3.2. Maschinelle Übersetzung

In der maschinellen Übersetzung sollen Texte a aus Sprache A in Texte b in Sprache B übersetzt werden. Im Rahmen des Noisy-Channel-Modells stellt die Originalnachricht m die gesuchte Übersetzung b dar und die vom Kanal veränderte Nachricht n den Quelltext a . Dies in Formel (2) angewandt ergibt die Grundlage der Herangehensweise bei maschineller Übersetzung:

$$b = \underset{b}{\operatorname{argmax}} P(a|b) \cdot P(b) \quad (3)$$

Dabei ist $P(a|b)$ das *Übersetzungsmodell*, welches bewertet wie gut der Quelltext a zu einer Übersetzung b passt. Das *Sprachmodell* $P(b)$ bewertet, wie gut der übersetzte Text b in der Sprache B ist. Beide Modelle zusammen bewerten eine Übersetzung b ; ein geeigneter Suchalgorithmus benutzt diese Bewertung um die beste Übersetzung zu finden.

Von mehreren Methoden ist die Wort-für-Wort Übersetzung die einfachste Strategie, bei der jedes Wort für sich sinngetreu übersetzt wird. Trotz oder gerade wegen dieser Einfachheit hat das Verfahren einige Schwächen. So gibt es manchmal einfach keine sinngetreuen Übersetzungen, andererseits müssen manche Worte der einen durch mehrere Worte in der anderen Sprache ausgedrückt werden und im schlimmsten Fall existieren ganze syntaktische Konstrukte in der anderen Sprache gar nicht. Trotzdem scheint sie mir wegen der Einfachheit und Ähnlichkeit als Vorbild für die Akronymgenerierung sinnvoll zu sein.

Grundlage der statistischen Übersetzung sind Sammlungen von Texten, über die statistische Daten gesammelt werden. Diese so genannten Korpora werden einmal in der Zielsprache benötigt um das Sprachmodell der Zielsprache zu bilden und als parallele Korpora mit inhaltlich identischen Texten in beiden Sprachen für das Übersetzungsmodell.

Eine dritte wichtige Komponente ist ein Algorithmus zur eigentlichen Dekodierung eines konkreten Textes. Eine simple Implementierung für die wortbasierte Übersetzung könnte zunächst alle Wörter mithilfe des Übersetzungsmodell in die wahrscheinlichste Entsprechung der Zielsprache übersetzen und anschließend die Wörter anhand des Sprachmodells der Zielsprache in die grammatikalisch richtige Reihenfolge bringen.

Quelle: [MAN03]

3.3. N-Gram-Modelle

Mit N-Gram-Modellen kann statistische Inferenz modelliert werden. Sie sind zum Beispiel geeignet, Sprachmodelle für die maschinelle Sprachverarbeitung zu bilden. Die Grundlage dafür ist, für eine gegebene Sequenz von Wörtern die Wahrscheinlichkeit des nächsten Wortes vorhersagen zu wollen. Also die Wahrscheinlichkeit des n -ten Wortes gegeben der vorangegangenen Worte im Satz $P(w_n|w_{n-1}, w_{n-2}, \dots, w_1)$.

N-Gram-Modelle machen hierbei die Markov-Annahme, dass die Wahrscheinlichkeit des n -ten Wortes nur von den $n-1$ vorangegangenen abhängig ist. Die Wahrscheinlichkeit eines Satzes mit m Wörtern kann dann als

$$P(w_1, \dots, w_m) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_2, w_1) \cdot \dots \cdot P(w_m|w_{m-1}, \dots, w_{m-n+1}) \quad (4)$$

berechnet werden. Die bedingte Wahrscheinlichkeit lässt sich errechnen aus:

$$P(w_m|w_{m-1}, \dots, w_2, w_1) = \frac{P(w_1, w_2, \dots, w_m)}{P(w_1, w_2, \dots, w_{m-1})} \quad (5)$$

Die Wortfolgen werden aus einer Textsammlung gelernt, indem je nach Grad des N-Gram-Modells, das gemeinsame Auftauchen von Wörtern gezählt wird. Wahrscheinlichkeiten können mit einem Maximum Likelihood Schätzer wie folgt geschätzt werden:

$$P(w_1, w_2, \dots, w_m) = \frac{C(w_1, w_2, \dots, w_m)}{N} \quad \text{und} \quad (6)$$

$$P(w_m | w_{m-1}, \dots, w_2, w_1) = \frac{C(w_1, w_2, \dots, w_m)}{C(w_1, w_2, \dots, w_{m-1})}. \quad (7)$$

Dabei bezeichnet $C(w_1, w_2, \dots, w_m)$ die gezählten Beobachtungen der Wortfolge (w_1, w_2, \dots, w_m) in der Trainingsmenge, deren Mächtigkeit N ist.

Quelle: [MAN03]

4. Modell für Akronymgenerierung

Für die Akronymgenerierung benutze ich ein abgewandeltes Modell der wortbasierten maschinellen Übersetzung. Es besteht ebenfalls aus zwei Teilen, dem *Buchstabenzuordnungs-* und dem *Akronymmodell*, die der Idee her dem Übersetzungs- und dem Sprachmodell entsprechen. Allerdings basieren die Modelle hier auf *Buchstabenalignments* und nicht auf *Wortalignments*.

4.1. Was ist ein Akronym?

Ein Akronym ist laut Duden ein aus den Anfangsbuchstaben anderer Wörter gebildetes Wort. Ich fasse den Begriff hier aber etwas weiter, sodass alle Abkürzungen, die im weitesten Sinne aussprechbar sind, als Akronyme gelten.

<i>Akronym</i>	<i>Bedeutung</i>
<i>ACID</i>	A tomicity, C onsistency, I solation, and D urability
<i>SOAP</i>	S imple O bject A ccess P rotocol
<i>TIGER</i>	T opologically I ntegrated G eographic E ncoding and R eferencing System
<i>CEBIT</i>	C entrum der B üro- und I nformationstechnik
<i>FIFA</i>	F édération I nternationale de F ootball A ssociation
<i>LAN</i>	L ocal A rea N etwork
<i>RADAR</i>	R adio D etection and R anging

Tabelle 1: Beispiele für Akronyme

Eine besondere Teilmenge der Akronyme sind solche die selber ein Wort sind. In Tabelle 1 sind das die Akronyme in den ersten drei Zeilen, aber auch *RADAR*, denn als Kriterium nehme ich das Vorkommen im Wörterbuch. Weiterhin zeigen die Beispiele, dass nicht nur Anfangsbuchstaben im Akronym vorkommen, wie etwa im Akronym *RADAR* oder *CEBIT*; es können auch mehrere Buchstaben aus einem Wort oder aber ein Wort gar nicht im Akronym vertreten sein.

Neben allgemeiner eigener Beobachtungen benutze ich die recht umfangreichen Listen der deutschen und englischen Wikipedia als Quellen für Akronyme:

- ◆ http://de.wikipedia.org/wiki/Liste_der_Akronyme und
- ◆ http://en.wikipedia.org/wiki/List_of_acronyms_and_initialisms.

Ich möchte nun einige Symbole einführen, die ich im Folgenden benutzen werde. Die Menge der Ausgangs- oder Quellwörter also die Wörter die durch das Akronym abgekürzt werden sollen bezeichne ich mit W . Deren Elemente w sind Wörter über das Alphabet X , das alle lateinischen

Buchstaben, sowie Leerzeichen, Kommas, Bindestrich und so weiter enthält. Das Alphabet der Akronyme Y besteht dagegen nur aus den kleinen lateinischen Buchstaben, ist also eine echte Teilmenge von X . Ich unterscheide bei den Akronymen nicht zwischen Groß- und Kleinschreibung, schreibe Akronyme aber zur besseren Erkennbarkeit immer in Großbuchstaben.

Zu allen Quellwörtern $w \in W$ gibt es eine Menge von Alignments A_w , deren Elemente a Vektoren über Indizes von w sind. Es gilt dabei, dass die Elemente in a paarweise verschieden und streng monoton wachsend sind, also

$$\forall a \in A_w: \forall i, j \in [1, \text{length}(a)]: i < j \rightarrow a_i < a_j, \quad (8)$$

denn jeder Buchstabe aus w soll höchstens einmal im Akronym vorkommen und die Reihenfolge der Buchstaben soll im Akronym erhalten bleiben. Außerdem müssen alle im Alignment indizierten Buchstaben der Quellwörter auch im Alphabet Y liegen, es darf also weder ein Leerzeichen noch ein Bindestrich indiziert werden. Folglich ist die maximale Länge eines Alignments zu einer Folge von Quellwörtern nach oben beschränkt durch die Anzahl der Buchstaben, die auch im Alphabet Y liegen.

$$\forall a \in A_w: \forall i \in [1, \text{length}(a)]: w(a_i) \in Y \quad (9)$$

$$\forall a \in A_w: \text{length}(a) \leq |\{i | w(i) \in Y\}| \quad (10)$$

Aus einem Alignment $a \in A_w$ und den Quellwörtern $w \in W$ erhält man ein Akronym in Textform indem der Reihe nach aus den Indizes des Alignments die entsprechenden Buchstaben der Quellwörter aneinander gereiht werden. Genau dies tut auch die Funktion $\text{getAcronym}: A_w \times W \rightarrow Y^*$ die wie folgt definiert ist:

$$\text{getAcronym}(a, w) = \bigwedge_{i=1}^{\text{length}(a)} w(a_i). \quad (11)$$

<i>Symbol</i>	<i>Bedeutung</i>
W	Menge aller Quellwörter
A_w	Menge aller Alignments zu den Quellwörtern w
X, Y	Alphabet der Quellwörter und der Akronyme, $Y \subset X, Y = \{ 'a', 'b', \dots, 'z', 'ä', 'ö', 'ü', 'ß' \}$
getAcronym	Funktion, die zu einem Alignment und den zugehörigen Quellwörtern das Akronym berechnet

Tabelle 2: Übersicht Symbole

Ich werde im Rest dieser Arbeit die Begriffe Akronym und Alignment synonym verwenden, meine aber immer das Alignment und damit die Buchstabenzuordnung. Bei der Generierung von Akronymen ist die Generierung von Alignments gemeint, deren textuelle Form das Akronym ist. Ein Akronym kann dabei durchaus mehrere Alignments haben, wie auch das folgende Beispiel zeigt.

Beispiel

Das Akronym *RADAR* steht für „Radio Detection And Ranging“. Die Quellwörter sind also $w = \text{Radio Detection And Ranging}$. Ein mögliches Alignment ist $a = (3, 4, 8, 9, 10, 21)$, das Akronym $\text{getAcronym}(w, a) = \text{dieter}$. Allerdings scheint dies kein gutes Akronym zu sein, da es relativ schwer ist, allein aus dem Akronym auf die Quellwörter zu schließen.

Ein nicht erlaubtes Alignment ist $(3, 6, 8, 9, 10, 21)$, da an der sechsten Stelle der Quellwörter ein Leerzeichen steht, welches nicht im Akronym vorkommen darf. Auch $(3, 13, 8, 9, 10, 21)$ ist wegen der Verletzung des monotonen Wachstums nicht erlaubt.

Das Akronym *RADAR* kann durch die Alignments $a^1=(1,2,3,16,21)$ und $a^2=(1,2,7,16,21)$ repräsentiert werden. Allerdings scheint die zweite Variante besser zu sein, da der dritte Buchstaben nicht der dritte Buchstabe eines Wortes, sondern Anfangsbuchstabe von „Detection“ ist.

4.2. Modell

Wir suchen zu gegebenen Quellwörtern w das beste Alignment $\hat{a} \in A_w$. Die angepasste Formel für die Suche ist die folgende:

$$\hat{a} = \underset{a \in A_w}{\operatorname{argmax}} P(\operatorname{getAcronym}(a, w)) \cdot P(w|a) \quad (12)$$

Dabei heißt $P(\operatorname{getAcronym}(a, w))$ Akronymmodell, $P(w|a)$ Buchstabenzuordnungsmodell. Das Akronymmodell bewertet wie gut ein Akronym hinsichtlich der Aussprechbarkeit ist, sollte also *RADAR* höher bewerten als *RDR*, das Buchstabenzuordnungsmodell bewertet wie wahrscheinlich die Quellwörter sind gegeben das Akronym; beim Beispiel Radar sollte also das Alignment höher bewertet werden, bei dem das „d“ der Anfangsbuchstabe des zweiten Wortes ist.

4.3. Akronymmodell

Das Akronymmodell bewertet eine Zeichenkette auf Aussprechbarkeit. Zwei Möglichkeiten der Realisierung möchte ich untersuchen, das Wörterbuch- und das N-Gram-Modell, wobei ersteres für die Realwort-Akronymgenerierung angewendet werden soll und letzteres für die Scheinwort-Akronymgenerierung.

4.3.1. Wörterbuchmodell

Um Akronyme zu finden, die auch reale Wörter sind, bietet sich die Benutzung eines Wörterbuchs an, wodurch die Berechnung der Wahrscheinlichkeit recht einfach ausfällt.

$$P(k) = \begin{cases} \frac{1}{N} & \text{falls } k \in \text{Wörterbuch} \\ 0 & \text{sonst} \end{cases} \quad (13)$$

Dabei ist N die Anzahl der Wörter im Wörterbuch und k ein Wort. Da bei der Suche nach dem Alignment mit maximaler Wahrscheinlichkeit gesucht wird und die Wahrscheinlichkeit des Wörterbuchmodells bei allen Akronymen gleich ist, beziehe ich sie nicht in die Berechnung mit ein. Ein Alignment fällt einfach aus der Suche raus, wenn es nicht im Wörterbuch ist.

4.3.2. N-Gram-Modell

Das Wörterbuchmodell ist jedoch sehr eingeschränkt hinsichtlich der Akronyme, die mit seiner Nutzung generiert werden können. Da alle Akronyme mit Wahrscheinlichkeit 0 bewertet werden, die keine echten Wörter sind, aber ein Großteil der Akronyme künstlich sind, ist ein flexibleres Modell notwendig. Wie bereits beschrieben, können N-Gram-Modelle bewerten, wie gut Sequenzen zu einer Menge von Referenzsequenzen passen; also zum Beispiel wie gut ein deutscher Satz zu einer Trainingsmenge von deutschen Sätzen passt. Wenn die Wörter im Satz auch häufig in der Trainingsmenge vorkommen und dort auch noch in der gegebenen Reihenfolge, dann ist die Wahrscheinlichkeit hoch, dass es ein guter deutscher Satz ist (im Sinne von Rechtschreibung und Grammatik).

Dieser Ansatz läßt sich auf Buchstabensequenzen übertragen, wenn statt Sequenzen von Wörtern die Folge von Buchstaben bewertet werden soll. Die Trainingsmenge besteht dann statt aus Sätzen aus Wörtern. Diese stammen beispielsweise aus einem Wörterbuch und das N-Gram-Modell bewertet dann, wie gut ein neues Wort zu den gelernten passt. Dabei muss das zu bewertende Wort bzw. die Buchstabensequenz kein echtes Wort sein und kann trotzdem eine gute Bewertung erhalten, wenn es aus Buchstabensequenzen besteht, die auch häufig in der Trainingsmenge anzutreffen sind.

Sei k nun ein Wort der Länge l bestehend aus Buchstaben k_1 bis k_l . Die Wahrscheinlichkeit des Wortes ist dann $P(k) = P(k_1, k_2, \dots, k_l)$. Für ein N-Gram-Modell n -ten Grades wird dabei angenommen, dass die Wahrscheinlichkeit eines Buchstaben nur von den $n-1$ vorangegangenen abhängig ist. Somit gilt

$$P(k_1, k_2, \dots, k_l) = P(k_1) \cdot P(k_2|k_1) \cdot P(k_3|k_2, k_1) \cdot \dots \cdot P(k_l|k_{l-1}, \dots, k_{l-n+1}) . \quad (14)$$

Dabei gilt wieder für $P(k_i|k_{i-1}, \dots, k_1)$

$$P(k_i|k_{i-1}, \dots, k_1) = \frac{P(k_1, k_2, \dots, k_{i-1}, k_i)}{P(k_1, k_2, \dots, k_{i-1})} \quad (15)$$

und die Wahrscheinlichkeit für eine Sequenz $P(k_1, k_2, \dots, k_i)$ wird mit einem Maximum Likelihood Schätzer berechnet:

$$P(k_1, k_2, \dots, k_i) = \frac{C(k_1, k_2, \dots, k_i)}{N} . \quad (16)$$

Die Trainingsmenge ist eine Liste von Wörtern, in denen die Buchstabenfolgen gezählt werden.

Zählvarianten

Für das Zählen möchte ich zwei verschiedene Varianten untersuchen, die sich darin unterscheiden, ob Buchstabensequenzen am Wortanfang gesondert gezählt werden. Dabei zählen N-Gramme, die den Buchstaben des Wortanfangs enthalten als N-Gramm am Wortanfang, wie in Abbildung 3 skizziert. Wenn die N-Gramme nach Wortanfangsposition unterschieden werden sollen, wird die Zählfunktion entsprechend angepasst als $C(k_1, k_2, \dots, k_i, \text{istWortanfang})$, wobei *istWortanfang* wahr oder falsch sein kann.

Ich erhoffe von der Einbeziehung der Unterscheidung, ob ein N-Gram am Wortanfang gesehen wurde, eine Verbesserung der Qualität der Bewertung. Den Grund möchte ich am Beispiel von Abbildung 3 illustrieren. Angenommen die Trainingsmenge würde nur aus dem Wort *sehen* bestehen und das Akronym *EHEN* soll bewertet werden. Dann würde dieses ohne Beachtung des Wortanfangs positiv beschieden werden, da diese Buchstabensequenz im Trainingskorpus beobachtet wurde. Andererseits würde es mit Beachtung des Wortanfangs keine gute Bewertung erhalten, da ja im Korpus kein Wort vorkommt, das mit *EHEN* beginnt. Mit Beachtung des Wortanfangs würde das N-Gram-Modell also etwas stärker dem oben beschriebenen Wörterbuchmodell gleichen.

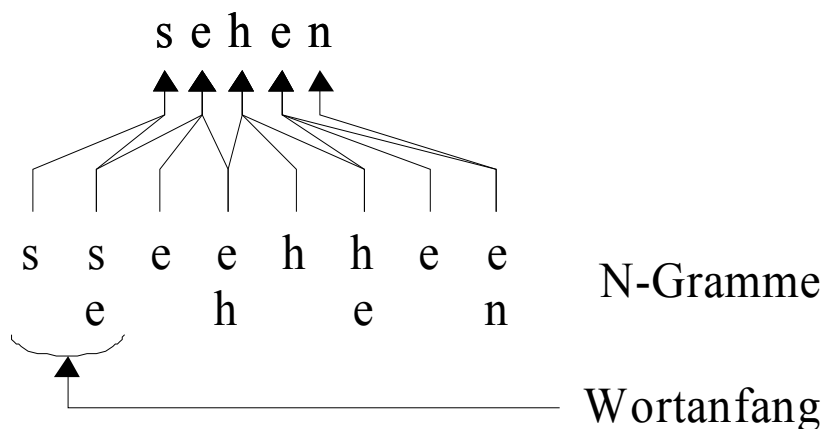


Abbildung 3: Zählung für ein 2-Gram-Modell mit Berücksichtigung des Wortanfangs

Normierung

Da für jeden weiteren Buchstaben des Akronymes in (14) ein neuer Faktor hinzukommt, wird die Wahrscheinlichkeit für längere Wörter immer kleiner sein als für kürzere, obwohl sie vielleicht

ähnlich gut aussprechbar sind. Für eine Art Normierung berechne ich die l -te Wurzel, wobei l die Anzahl der Buchstaben ist:

$$P_{\text{normiert}}(k_1, k_2, \dots, k_l) = (P(k_1, k_2, \dots, k_l))^{\frac{1}{l}} \quad (17)$$

Die Wahrscheinlichkeit ist ein Produkt über l Faktoren, also:

$$P(k_1, k_2, \dots, k_l) = P(k_1) \cdot P(k_2|k_1) \cdot \dots \cdot P(k_l|k_{l-1}, \dots, k_{l-n+1}). \quad (18)$$

Da ich mit Logarithmen rechne, gilt

$$\ln(P(k_1, k_2, \dots, k_l)) = \ln(P(k_1)) + \ln(P(k_2|k_1)) + \dots + \ln(P(k_l|k_{l-1}, \dots, k_{l-n+1})) \quad (19)$$

und für den normierten Wert:

$$\ln(P_{\text{normiert}}(k_1, k_2, \dots, k_l)) = \frac{1}{l} \cdot \ln(P(k_1, k_2, \dots, k_l)). \quad (20)$$

Ob diese Normierung zu einem verbesserten Modell führt, wird sich zeigen müssen, sie führt aber jedenfalls auch zu einer Angleichung an das oben beschriebene Wörterbuchmodell, da dort alle Worte unabhängig von der Länge gleich bewertet werden, sofern sie nur im Wörterbuch stehen.

4.4. Buchstabenzuordnungsmodell

Das Buchstabenzuordnungsmodell bewertet die Güte eines Alignments a hinsichtlich der Quellwörter w , wobei a ein Vektor über Indizes von Buchstaben in w ist. Zwei Attribute der Buchstaben sollen hier in die Bewertung der Güte des Alignments eingehen, das ist zum einen die Position des Buchstaben im Wort und die Information, ob das Wort ein Stoppwort ist.

Diese Attribute habe ich gewählt, weil bei der Analyse von Akronymbeispielen meist die Anfangsbuchstaben für das Akronym benutzt werden und Wörter ohne im Akronym vorkommenden Buchstaben zumeist Stoppwörter sind. Stoppwörter sind dabei sehr häufig vorkommende Wörter, die weniger bedeutungstragende als eher funktionelle Bedeutung haben, also wie etwa „und“, „oder“, Pronomen und Artikel.

Das Buchstabenzuordnungsmodell setzt sich also als Produkt der Wahrscheinlichkeiten hinsichtlich der beiden Attribute zusammen, also als

$$P(w|a) = P_{\text{wordpos}}(w|a) \cdot P_{\text{stopword}}(w|a). \quad (21)$$

Die Faktoren könnten auch noch gewichtet werden, was ich allerdings im Rahmen dieser Arbeit nicht untersuche. Lediglich die Performanz mit und ohne den Faktor P_{stopword} werde ich messen.

Attribut Wortposition

Sei nun M die Menge der Anfangsbuchstabenindizes der Wörter in w , also $M = \{n | n \in \mathbb{N} \wedge 1 \leq n \leq \text{length}(w) \wedge \text{isLetter}(w(n)) \wedge \neg \text{isLetter}(w(n-1))\}$. Das Attribut Wortposition ist eine Funktion $\text{wordpos} : W \times \mathbb{N} \rightarrow \mathbb{N}$ mit

$$\text{wordpos}(w, i) = \begin{cases} 0, & \text{falls } \neg \text{isLetter}(w(i)) \vee i < 1 \vee i > \text{length}(w) \\ i - j + 1, & \text{mit } j = \min(\{m | m \in M \wedge m \leq i\}) \end{cases}. \quad (22)$$

Die Wahrscheinlichkeit der Quellwörter w gegeben das Alignment a ist dann

$$P_{\text{wordpos}}(w|a) = \prod_{\substack{i \in [1, \dots, \text{length}(w)] \\ i \in a}} P(\text{true} | \text{wordpos}(w, i)) \cdot \prod_{\substack{i \in [1, \dots, \text{length}(w)] \\ i \notin a}} P(\text{false} | \text{wordpos}(w, i)). \quad (23)$$

Die beiden Verteilungen für die Klassen *true* und *false* werden aus einer Trainingsmenge mit Quellwörtern und Buchstabenzuordnungen (Alignments) gelernt. *True* steht dabei für die Klasse von Buchstabenindizes, die in einer Buchstabenzuordnung vorkommen, *false* für die Komplementärklasse. $C(\text{true}|n)$ steht für die Anzahl von Beispielen, bei denen der n -te Buchstabe eines Wortes in der Zuordnung vorkommt, $C(\text{false}|n)$ analog dafür, dass der n -te Buchstabe nicht

in der Zuordnung gefunden wurde. In der Summe sollten beide alle Beobachtungen eines n-ten Buchstaben ergeben. Als Schätzer ergeben sich

$$P(\text{true}|n) = \frac{C(\text{true}|n)}{C(\text{true}|n) + C(\text{false}|n)}$$

und analog

$$P(\text{false}|n) = \frac{C(\text{false}|n)}{C(\text{true}|n) + C(\text{false}|n)}.$$

Attribut Stoppwort

Sei $\text{words}(w)$ die Menge aller Wörter in w , also zusammenhängende Zeichenketten die nur aus Buchstaben aus dem Alphabet Y bestehen und S die Menge aller Stoppwörter. Das Attribut Stoppwort ist wieder in Klassen true und false geteilt, jedoch auf der Ebene von Wörtern. Falls in einem Wort einer der Buchstaben liegt, die im Alignment indiziert sind, so ist es in der Klasse true .

Die Wahrscheinlichkeit der Quellwörter w gegeben das Alignment a für das Attribut Stoppwort ist dann

$$P_{\text{stopword}}(w|a) = \prod_{\substack{v \in \text{words}(w) \\ \exists j: a_j \in v}} P(\text{true}|v \in S) \cdot \prod_{\substack{v \in \text{words}(w) \\ \forall j: a_j \notin v}} P(\text{false}|v \in S). \quad (24)$$

Die Verteilungen werden auch hier aus den Trainingsbeispielen anhand der Zahlen $C(\text{true}|s)$ und $C(\text{false}|s)$ geschätzt, wobei s für ein Stoppwort wahr und sonst falsch ist.

Die Schätzer sind

$$P(\text{true}|s) = \frac{C(\text{true}|s)}{C(\text{true}|s) + C(\text{false}|s)} \quad (25)$$

und eben wieder

$$P(\text{false}|s) = \frac{C(\text{false}|s)}{C(\text{true}|s) + C(\text{false}|s)}. \quad (26)$$

4.5. Semantische Klassen

Es sollen keine negativen Akronyme generiert werden, bzw. der Wortlaut der Akronyme soll positiv sein. Aus diesem Grunde wird ein Mechanismus benötigt, mit dem solche unerwünschten Wörter erkannt werden können. Als Ausgangsbasis dient eine Liste mit Wörtern negativer Bedeutung, semantisch ähnliche Wörter lassen sich mithilfe eines Thesaurus' finden. Modellieren möchte ich die Ähnlichkeit mit einem Graphen, dessen Knoten Wörter sind und dessen Kanten ähnliche Wörter verbinden. Auf diese Weise lassen sich zu einem Wort alle semantisch ähnlichen Worte anhand des Graphen finden und auch eine Art Transitivität modellieren. Das heißt, dass Wörter die nicht zusammen im Thesaurus stehen, aber zu denen ein drittes Wort existiert, dass auf beide verweist, im Graphen mit einem Pfad über dieses dritte Wort verbunden sind. Dadurch bilden sich Klassen semantisch ähnlicher Wörter, die im Graphen untereinander erreichbar sind. Die semantische Ähnlichkeit ist dann der Abstand im Graphen.

Sei nun G ein ungerichteter Graph $G = (V, E)$ mit $E \subseteq V^2$, dessen Knoten V Worte sind und alle Kanten $e = \{v_1, v_2\}$ besitzt, bei denen v_2 im Thesaurus ein Synonym von v_1 ist.

Semantische Ähnlichkeit lässt sich nun als Abstand innerhalb des Graphen G ausdrücken. Die direkte Nachbarschaft $N(v) = \{u \in V | (v, u) \in E\}$ enthält die Wörter, die dem Wort v am ähnlichsten sind und sollte die Worte enthalten, die auch beim Nachschlagen im Thesaurus herausgekommen wären. Diese Wörter sind von semantischer Ähnlichkeit ersten Grades oder eben Nachbarn ersten Grades.

Mengen von Wörter mit höheren semantischen Grad werden durch Iteration der Nachbarschaft berechnet, etwa als $N_i(v) = \bigcup_{u \in N_{i-1}} N(u)$. Mit diesem Graphen G und einer Liste negativer Wörter lassen sich nun diese und semantisch ähnliche Wörter aus dem Wörterbuch entfernen, sodass keine „negativen“ Akronyme generiert werden können. Solche Wörterbücher heißen im Folgenden immer *bereinigte* Wörterbücher.

5. Dekodierung

Für die Dekodierung benutze ich zwei leicht verschiedene Verfahren, wobei das eine für die Realwort-Generierung geeignet ist, das andere für die Scheinwort-Akronymgenerierung. Beide basieren jedoch auf einer Tiefensuche über Buchstabenalignments.

Ein Stack speichert den aktuellen Zustand der Suche und wird mit einem Alignment für jeden Buchstaben der Quellwörter initialisiert. Jedes dieser initialen Alignments enthält nur den Index seines Buchstaben. Für die Suche wird das oberste Alignment vom Stack genommen und neue Alignments generiert. Ein neues Alignment wird für jeden Index der Quellwörter generiert, der größer ist als der letzte Index des Alignments, indem an das alte Alignment genau einer dieser Indizes angehängen wird. Auf diese Weise können alle möglichen Alignments generiert werden.

Eine kombinatorische Explosion wird durch wirksames Pruning, also dem Abschneiden des Suchbaumes verhindert. Bei der Realwort-Akronymgenerierung wird dafür ein Wörterbuch benutzt. Wird ein neues Alignment gebildet, so wird im Wörterbuch nachgeschaut, ob ein Wort existiert, dass mit dem Alignment beginnt. Ist dies nicht der Fall, können dieses Alignment und alle die später in dem Suchast noch generiert werden würden, keine gültigen Realwort-Akronyme mehr werden und ein Abschneiden des Suchastes ist gerechtfertigt.

Bei der Scheinwort-Generierung wird die Wahrscheinlichkeit des Alignments berechnet und mit einem Grenzwert verglichen, der wenn er unterschritten wird, zum Abschneiden des jeweiligen Astes führt. Dies passiert unter der Annahme, dass sich die Wahrscheinlichkeit von Alignments nur verschlechtern kann, wenn zu diesem weitere Indizes hinzu kommen. Diese Annahme ist leider etwas problematisch, da es durchaus auch sein könnte, dass ein längeres Alignment vor allem im Falle von Normierung in der Wahrscheinlichkeit besser werden kann. Ein möglichst spätes Abschneiden würde man durch einen niedrigeren Grenzwert erreichen. Andererseits wird dadurch der zu durchsuchende Raum größer, die Suche dauert länger.

Für die folgenden Pseudocode-Abschnitte benutze ich die üblichen Bezeichnungen. Weitere soll die folgende Tabelle 3 erklären.

<i>Symbol</i>	<i>Bedeutung</i>
<code>isLetter(c)</code>	Funktion, die prüft, ob ein Zeichen ein Buchstabe ist; $isLetter(c) = c \in Y$
<code>alignment</code>	Wie bisher ein Vektor von Indizes, über <code>add</code> kann ein weiterer Index an ein Alignment angefügt werden, <code>new alignment</code> ergibt ein leeres Alignment
<code>Dictionary</code>	Eine Liste deren Elemente die Wörter des Wörterbuchs sind
<code>length(w)</code>	Gibt die Anzahl der Zeichen einer Zeichenkette, oder die Zahl der Elemente eines Alignments zurück
<code>getAcronym(w, a)</code>	Funktion, die zu einem Alignment und den zugehörigen Quellwörtern das Akronym berechnet
<code>existsWord(D, b)</code>	Funktion, die bestimmt ob die Zeichenfolge <code>b</code> ein Element von <code>D</code> ist
<code>existsWord startingWith (D, b)</code>	Funktion, die bestimmt ob in <code>D</code> ein Element existiert, deren Anfangsbuchstaben mit <code>b</code> übereinstimmen

<i>Symbol</i>	<i>Bedeutung</i>
$S.insert(b)$ by A	Fügt in die Liste bzw. den Stack S das Element b an die Stelle ein, in die es bei Sortierung nach Attribut A hingehört
$P(a, w, M, N)$	Berechnet die Wahrscheinlichkeit des Alignments a hinsichtlich der Quellwörter w , mit Akronymmodell M und Buchstabenzuordnungsmodell N

Tabelle 3: Übersicht Symbole Dekodierung

5.1. Suche nach echten Wörtern

Ich benutze eine Tiefensuche mit einem Wörterbuch für die Suche nach Realwort-Akronymen. Mithilfe dieses Wörterbuchs lassen sich die beiden Funktionen `existsWord` und `existsWordStartingWith` realisieren, die als Eingabe eine Zeichenkette akzeptieren und prüfen, ob ein Wort im Wörterbuch steht, dass mit dem Argument übereinstimmt bzw. zumindest mit der Zeichenfolge beginnt. Die Funktion `getAcronym` setzt einfach ein gegebenes Alignment in eine gegebene Quellwortzeichenkette ein und liefert eine textuelle Repräsentation des Akronyms, die dann gegen das Wörterbuch verglichen werden kann.

Im Folgenden ist der Pseudocode für die Suche nach Akronymen mit echten Wörtern aufgelistet.

```

1  function wordSearch
2  input: w - String, D - dictionary,
3         N - LetterAlignmentModel
4  output: G - sorted List of (alignment, probability)
5
6  M <- new wordmodel(D)
7  G <- new List of (alignment, probability)
8  S <- new Stack of alignment
9
10 (init)
11 for i = length(w) downto 1
12     if (isLetter(w[i]))
13         a <- new alignment
14         a.add(i)
15         S.push(a)
16
17 (search)
18 while not isEmpty(S)
19     a <- S.pop
20     for i = length(w) downto (a[length(a)] + 1)
21         if (isLetter(w[i]))
22             a' <- a.add(i)
23             b <- getAcronym(w, a')
24             if (existsWordStartingWith(D, b))
25                 S.push(a')
26                 if (existsWord(D, b))
27                     p <- P(a', w, M, N)
28                     G.add((a', p))
29
30 (finish)
31 sort(G) by probability
32 return G

```

Die Suche liefert eine Menge von Alignment-Wahrscheinlichkeits-Paaren sortiert nach der Wahrscheinlichkeit. Wegen der effektiven Einschränkung des Suchraums aufgrund des Prunings

anhand der Wörterbuchbeschränkung ist eine erschöpfende Suche möglich, die gespeicherte Ergebnismenge kann aber auch leicht auf die 100 oder das einzige beste Alignment eingeschränkt werden.

Beispiel

Die Realwort-Akronymgenerierung möchte ich anhand der Quellwörter *Der tolle rote Wagen* skizzieren, die eher beliebig sind und für die kein Akronym existiert. Diese Quellwörter werden auch später in den Tests als Beispiel genutzt. Sie sind auch deshalb interessant, weil alle Anfangsbuchstaben der Worte Konsonanten sind, eine einfache Aneinanderreihung dieser also kein aussprechbares Akronym ergibt. Die Indizes der Buchstaben sind wie folgt:

Wörter: D e r t o l l e r o t e W a g e n
 Indizes: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Initialisierung: Zunächst werden in den Zeilen zehn bis 15 für alle Buchstaben der Quellwörter ein Alignment erstellt, das genau diesen Buchstaben enthält. Diese werden in umgekehrter Reihenfolge in den Stack gespeichert, sodass dasjenige Alignment oben liegt, welches aus dem ersten Buchstaben besteht. Der Stack sieht dann so aus: $S = ((1), (2), (3), (5), (6), \dots, (19), (20))$.

Suche: Die Suche (Zeilen 17 bis 28) expandiert in jedem Schritt das oberste Element des Stacks. Im ersten Schritt wird $a = (1)$ vom Stack genommen. In der Schleife (Zeile 20) wird für jede Iteration dieses Alignment um genau einen Buchstaben erweitert und geprüft ob die textuelle Form des neuen Alignments als Anfang eines Wortes im Wörterbuch vorkommt (Zeile 24). Das erste erweiterte Alignment ist $a' = (1, 20)$ und dessen textuelle Form *DN* ist nicht Wortanfang eines Wortes im Wörterbuch, somit lohnt es sich nicht weiter nach Akronymen zu suchen die mit *DN* beginnen und $a' = (1, 20)$ fällt aus der Suche raus. Die textuelle Form *DE* des nächsten Kandidaten $a' = (1, 19)$ ist Wortanfang vieler Worte und landet deshalb wieder auf dem Stack. Die Prüfung, ob es auch ein Wort ist (Zeile 26) wird wohl fehlschlagen, deshalb wird es nicht zur Ergebnismenge hinzugefügt². Das letzte Alignment, das auch wieder auf dem Stack landet ist $a' = (1, 2)$.

In der nächsten Iteration der großen Schleife (Zeile 18) wird $a = (1, 2)$ vom Stack genommen und wieder erweitert. Gleich die erste Erweiterung $a' = (1, 2, 20)$ (textuell *DEN*) ist ein Wort, dessen Wahrscheinlichkeit berechnet und in der Ergebnismenge gespeichert wird. Weitere Alignments die in die Ergebnismenge gelangen sind $a' = (1, 2, 16)$ (*DEW*, engl.: Tau), $a' = (1, 2, 11)$ (*DER*) und $a' = (1, 2, 3)$ (*DER*)³. Dieses $a = (1, 2, 3)$ wird in der nächsten Iteration erweitert und so weiter.

Abschluss: Zum Schluss wird die Ergebnismenge aufsteigend nach den Wahrscheinlichkeiten sortiert.

5.2. Suche nach Scheinwörtern

Für die allgemeinere Suche nach Akronymen, die auch Scheinwörter sein dürfen, wird ein N-Gram-Modell benötigt, das die Aussprechbarkeit oder Ähnlichkeit zu echten Wörtern modelliert. Der Suchzustand wird in einer Liste gespeichert, die diesmal Paare von Alignments und deren Wahrscheinlichkeiten speichert, die sortiert ist und auf deren Elemente über einen Index zugegriffen werden kann. Gleichzeitig gibt es eine lokale Variable `threshold`, welche die minimale Wahrscheinlichkeit speichert, die ein Alignment haben muss, um in die Suchzustand-Liste zu gelangen. Dieser Wert ist anfangs sehr klein. Das Suchergebnis wird in der sogenannten Zielliste gespeichert, welche die besten bisher gefundenen Alignments aufbewahrt. Die Breite der Suche wird über einen weiteren ganzzahligen nichtnegativen Parameter bestimmt. Ist die maximale Breite erreicht und enthält die Zielliste mehr Alignments als die Breite spezifiziert, so wird die Zielliste

² Dies könnte aber später passieren, wenn das Alignment auf $a' = (1, 19, 20)$ erweitert wird, denn *den* ist ein Wort, dessen Wahrscheinlichkeit allerdings gering ist.

³ Leider wird auch $a' = (1, 2, 18)$ (*DEG*, engl. Abkürzung für degree) in die Ergebnismenge eingefügt, weil das englische Wörterbuch leider noch Abkürzungen enthält.

verkleinert und `threshold` auf die Wahrscheinlichkeit des schlechtesten Alignments in der Zielliste gesetzt.

Es folgt der Pseudocode für die Suche nach Akronymen mit Scheinwörtern.

```
1      function ngramSearch
2      input: w - String, M - ngram-model, t - integer
3             N - LetterAlignmentModel
4      output: G - sorted List of (alignment, probability)
5
6      G <- new List of (alignment, probability)
7      S <- new List of (alignment, probability)
8      threshold <- very low probability
9
10     (init)
11     for i = length(w) downto 1
12         if (isLetter(w[i]))
13             a <- new Alignment
14             a.add(i)
15             p <- P(a, w, M, N)
16             S.add((a, p))
17     sort(S) by probability
18
19     (search)
20     while not isEmpty(S)
21         (a, p) <- S.getBestProbability
22         for i = length(w) downto (a[length(a)] + 1)
23             if (isLetter(w[i]))
24                 a' <- a.add(i)
25                 p <- P(a', w, M, N)
26                 if (p >= threshold)
27                     S.add((a', p))
28         sort(S) by probability
29         G.insert((a, p)) by probability
30         if (length(G) > t)
31             removeLastElement(G)
32             (a2, p2) <- lastElement(G)
33             threshold <- p2
34
35     (finish)
36     return G
```

Die Funktion `ngramSearch` liefert die t besten Alignments. Die Suche läuft hier so ähnlich wie bei den echten Wörtern ab, mit dem Unterschied dass nicht so direkt entschieden werden kann, wann ein Alignment nicht mehr gut ist und auch bei Erweiterung kein gutes Akronym mehr liefert. Deshalb wird wie bei einer Beam-Suche⁴ eine festgelegte Anzahl von besten Alignments gesammelt und Alignments, deren Wahrscheinlichkeit schlechter ist als die schlechteste Wahrscheinlichkeit der gesammelten Alignments, werden verworfen.

4 In einer normalen Bestensuche merkt sich der Algorithmus immer nur ein bestes Element. In der Expansionsphase wird von diesem besten Element das beste Nachfolge-Element bestimmt, welches dann zum neuen besten Element wird und von dem aus wieder der beste Nachfolger bestimmt wird und so weiter.
Bei der Beam-Suche mit der Breite m , werden m viele beste Elemente gespeichert, was die Wahrscheinlichkeit erhöht, wirklich gute Elemente zu finden.

6. Experimentelle Untersuchung

Wie gut die Algorithmen sind, habe ich mit den 854 englischen und 79 deutschen Akronymen untersucht. Dabei wird auf Seite des Buchstabenzuordnungsmodells auf einer Teilmenge der Akronyme trainiert und auf dem Rest das trainierte Modell getestet. Die Güte der Modelle soll sich an drei Faktoren messen: der Anzahl der korrekt generierten Akronyme, der Anzahl der gar nicht generierten Akronyme, der durchschnittlichen Log-Likelihood Differenz und der Rang-Abdeckungs-Kurve. Außerdem sollen die Charakteristika der verschiedenen Varianten bei der Generierung von Akronymen zu den bereits vorgestellten Quellwörtern `Der tolle rote Wagen` deutlich werden.

Aus der Anzahl der korrekt generierten Akronyme, läßt sich die *Fehlerrate* R berechnen als:

$$R = \frac{N - C}{N} . \quad (27)$$

Dabei ist N die Anzahl aller Testbeispiele und C die Anzahl der korrekt generierten. Als korrekt generiert zählen dabei alle Testdurchläufe, bei denen das erwartete Akronym die höchste Wahrscheinlichkeit von allen erhält, oder aber unter denen mit der höchsten Wahrscheinlichkeit ist, falls mehrere Akronyme mit gleicher Wahrscheinlichkeit den ersten Platz belegen.

Die Anzahl der gar nicht generierten Akronyme beschreibt, wieviele der erwarteten Akronyme gar nicht in der Ergebnismenge lagen, die also nicht generiert wurden. Daraus läßt sich auch der prozentuale Anteil T an der Gesamtmenge berechnen als

$$T = \frac{N - F}{N} . \quad (28)$$

Dabei ist wieder N die Anzahl aller Testbeispiele und F die Anzahl der Beispiele, bei denen das erwartete Alignment nicht generiert wurde.

Die *durchschnittliche Log-Likelihood Differenz* wird folgendermaßen berechnet. Für jedes Akronymbeispiel der Testmenge, bei der das erwartete Akronym nicht unter denen mit der höchsten Wahrscheinlichkeit ist, wird die Differenz der Log-Likelihoods zwischen dem oder den besten generierten Akronymen und dem erwarteten Akronym gebildet. Wenn erwartete Akronyme gar nicht generiert werden konnten, wurde auch keine Wahrscheinlichkeit für sie berechnet, sie zählen also hier nicht rein. Die Differenzen werden über die ganze Testmenge summiert und ergeben dann die kumulierte Log-Likelihood Differenz. Wird diese durch die Anzahl der fehlgeschlagenen Testbeispiele (ohne die, die gar nicht generiert wurden) dividiert, so ergibt sich durchschnittliche Log-Likelihood Differenz. Die durchschnittliche Log-Likelihood Differenz beschreibt also wie groß die Differenz zwischen den Wahrscheinlichkeiten von erwarteten und besten Alignments im Falle eines Fehlschlags durchschnittlich ist. Eine betragsmäßig kleinere Differenz ist dabei besser, da hier bei den einzelnen fehlgeschlagenen Testbeispielen durchschnittlich das Modell den erwarteten Akronymen eine höhere Wahrscheinlichkeit berechnete. Anders ausgedrückt ist die Wahrscheinlichkeits-Lücke vom erwarteten Akronym zur Spitze kleiner.

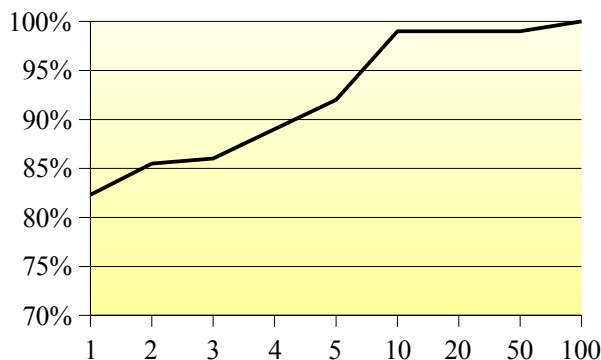


Abbildung 4: Beispiel Rang-Abdeckung-Kurve

Der Rang der erwarteten Akronyme in der gesamten Testmenge wird in der *Rang-Abdeckungs-Kurve* abgebildet. Auf der Abszisse ist der Platz abgezeichnet und auf der Ordinate der Anteil der Testbeispiele deren erwartetes Akronym sich unter dem ersten bis zu diesen Platz befindet. Die Kurve ist immer monoton wachsend und schneidet die Ordinate bei dem Anteil der Testbeispiele, bei denen das erwartete Alignment auf dem ersten Platz ist, also das Akronym erfolgreich generiert wurde. Ein Beispiel ist in Abbildung 4 dargestellt. Es liegen hier 86 Prozent der erwarteten Alignments entweder auf Platz eins oder zwei und 99 Prozent unter den ersten zehn Plätzen. Die Fehlerrate kann auch abgelesen werden, sie ist der fehlende Anteil zu 100 Prozent beim Wert 1 auf der Abszisse, hier also rund 17 Prozent. Für den Rang gilt die Reihenfolge nach der Wahrscheinlichkeit. Wenn also mehrere Alignments die gleiche Wahrscheinlichkeit haben, so haben sie den gleichen Rang in der Bestenliste; alle Alignments mit der nächstschlechteren Wahrscheinlichkeit haben auch den nächstschlechteren Rang.

Für die vielen Varianten der Scheinwort-Akronymgenerierung gibt es ein Balkendiagramm mit dem Anteil der erfolgreich generierten Akronyme und eins mit dem Anteil total fehlgeschlagener Generierung. Wie bisher zählt ein Akronym für ein Testbeispiel (das aus Quellwörtern und einem Alignment besteht) als *erfolgreich* generiert, wenn das vorgegebene Alignment des Beispiels unter denen mit der höchsten Wahrscheinlichkeit ist. Als *total fehlgeschlagen* gilt eine Generierung für ein Beispiel, wenn das gegebene Alignment des Beispiels nicht unter den 100 besten generierten Alignments ist oder gar nicht generiert wurde. Der Begriff *total fehlgeschlagen* ist bitte nicht mit dem normalen fehlgeschlagen zu verwechseln, denn eine Generierung gilt schon als „normal“ fehlgeschlagen, wenn sie nicht erfolgreich ist.

Die Quellen von deutschen und englischen Wörterbüchern und Thesauren sind folgende:

- ◆ http://lingucomponent.openoffice.org/spell_dic.html,
- ◆ http://lingucomponent.openoffice.org/thes_dic.html und
- ◆ <http://www.openthesaurus.de>.

Für die Beseitigung von negativen Wörtern habe ich aus der deutschen Wortliste eine Liste von etwa 2000 von mir als negative eingestufte Wörter erstellt, diese anschließend ins Englische übersetzt. Für diese Übersetzung habe ich die Übersetzungsliste der TU Chemnitz eingesetzt, die unter

- ◆ <http://dict.tu-chemnitz.de/>

erhältlich ist. Als Wörterbuch für die Akronymgenerierung kann eines der beiden einzelnen oder die Vereinigung beider benutzt werden. Wird im Folgenden von bereinigten Wörterbüchern gesprochen, so sind die negativen Wörter und deren Synonyme ersten Grades entfernt worden.

6.1. Akronymgenerierung echter Wörter

Das Buchstabenzuordnungsmodell wird für diese Experimente auf den Akronymbeispielen trainiert, die keine echten Wörter sind, wohingegen auf den Realwort-Akronymen getestet wird. Ich teste verschiedene Szenarien, die sich im verwendeten Wörterbuch (nur Englisch oder gemischt Deutsch und Englisch), der Säuberung von schlechten Wörtern und den verwendeten Attributen im Buchstabenzuordnungsmodell unterscheiden.

<i>Sprache</i>	<i>Attribute</i>	<i>Korrekt generierte Akronyme</i>	<i>Fehlerrate</i>	<i>Durchschnittliche Log-Likelihood Differenz</i>
Englisch	Position	150 von 175	14,29%	3,2882
	Position und Stoppwörter	149 von 175	14,86%	3,3574
Deutsch und Englisch	Position	175 von 209	16,27%	2,9935 ⁵
	Position und Stoppwörter	174 von 209	16,75%	3,1125 ⁵

Tabelle 4: Ergebnis Suche Realwort-Akronyme, bereinigt

In Tabelle 4 ist das Ergebnis des Tests für bereinigte Wörterbücher aufgelistet. Die erste Spalte der Tabelle gibt die Sprache des benutzten Wörterbuchs an, die zweite die benutzten Attribute für das Buchstabenzuordnungsmodells, die dritte Spalte führt die Zahl der erfolgreich generierten Akronyme auf, in Spalte vier steht der prozentuale Anteil der fehlerhaft generierten Akronyme und die letzte Spalte zeigt die durchschnittliche Log-Likelihood Differenz.

<i>Sprache</i>	<i>Attribute</i>	<i>Korrekt generierte Akronyme</i>	<i>Fehlerrate</i>	<i>Durchschnittliche Log-Likelihood Differenz</i>
Englisch	Position	202 von 233	13,30%	3,0699
	Position und Stoppwörter	201 von 233	13,73%	3,1402
Deutsch und Englisch	Position	234 von 270	13,33%	3,2692
	Position und Stoppwörter	234 von 270	13,33%	3,4770

Tabelle 5: Ergebnis Suche Realwort-Akronyme, nicht bereinigt

Die Ergebnisse des Tests ohne Bereinigung des Wörterbuchs sind in Tabelle 5 dargestellt, die Rang-Abdeckungs-Kurven in Abbildung 5.

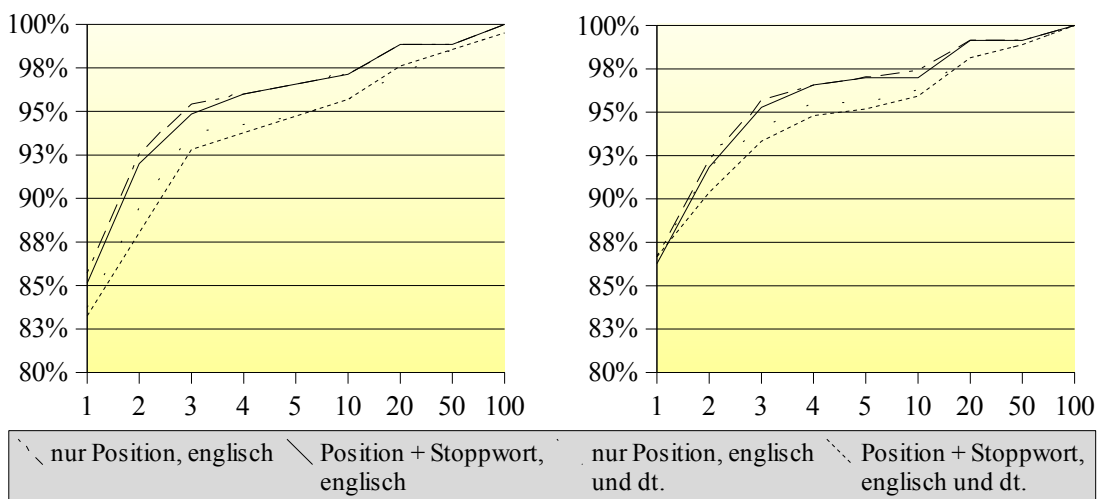


Abbildung 5: Rang-Abdeckungs-Kurven Realwort-Akronymgenerierung

Links: mit Bereinigung der Wörterbücher; Rechts: ohne Bereinigung

5 Ein Akronym nicht generiert: JATO (Jet-Assisted Take-Off).

Dem Ergebnis des Tests ist zu entnehmen, dass es rein von der Anzahl der richtig generierten Akronymen keinen Unterschied macht, ob das Attribut Stoppwort mit einberechnet wird; die durchschnittliche Log-Likelihood Differenz ist sogar leicht höher. Die Bereinigung des Wörterbuchs verschlechtert die Leistung des Modells leicht um ein bis drei Prozentpunkte, genauso wie die Vermischung von englischen und deutschen Wörterbuch.

Die Rang-Abdeckungs-Kurven zeigen, dass selbst im schlimmsten Fall 95 Prozent der generierten Akronyme unter den Top-Five sind.

In der folgenden Tabelle 6 sind die ersten zehn Alignments einer Realwort-Generierung mit nicht bereinigten englischen Wörterbuch und den schon bekannten Quellwörtern *Der tolle rote Wagen* aufgelistet. Die Ziffer beschreibt die Position in der Wahrscheinlichkeits-Rangliste; gleiche Position heißt, dass die Wahrscheinlichkeiten hier gleich sind. Der Artikel *Der* ist ein Stoppwort.

<i>Attribut: Position</i>	<i>Attribute: Position und Stoppwort</i>
TROW - Der tolle rote Wagen 1	TROW - Der tolle rote Wagen 1
DREW - Der tolle rote Wagen 2	TOW - Der tolle rote Wagen 2
ROW - Der tolle rote Wagen 3	TOW - Der tolle rote Wagen 2
TOW - Der tolle rote Wagen 3	TOR - Der tolle rote Wagen 2
TOW - Der tolle rote Wagen 3	DREW - Der tolle rote Wagen 3
TOR - Der tolle rote Wagen 3	TROT - Der tolle rote Wagen 4
DEW - Der tolle rote Wagen 3	TORT - Der tolle rote Wagen 4
DROWN - Der tolle rote Wagen 4	DEW - Der tolle rote Wagen 5
DETER - Der tolle rote Wagen 4	ROW - Der tolle rote Wagen 6
DRAG - Der tolle rote Wagen 5	DROWN - Der tolle rote Wagen 7

Tabelle 6: Beispiel Realwort-Akronymgenerierung mit Quellwörtern *Der tolle rote Wagen*, englisches Wörterbuch, nicht bereinigt

6.2. Akronymgenerierung von Scheinwörtern

Für die Suche nach Scheinwörtern werden für das Akronymmodell N-Gram-Modelle verwendet, die aus Wörterbüchern gelernt werden. Ich teste dabei Modelle verschiedenen Grades von zwei bis fünf aber auch verschiedene Berechnungsverfahren, wie der Normierung der Wahrscheinlichkeit und die zwei verschiedenen Zählvarianten bei den aufgetauchten Buchstabensequenzen. Als Wörterbuch habe ich hauptsächlich das englische benutzt und nur zum Vergleich in Tabelle 11 eine Generierung mit gemischtem Wörterbuch durchgeführt. Für das Buchstabenzuordnungsmodell gibt es die ebenfalls schon vorgestellten Varianten „nur Position“ und „Position mit Stoppwörtern“.

Der Test wird mit Kreuzvalidierung durchgeführt, denn zum Testen des Modells kommen diesmal alle Akronymbeispiele in Frage. Die Menge der Beispiele wird in zehn gleich große Partitionen geteilt und anschließend zehn Trainings- und Testprozesse durchlaufen. Insgesamt ist dabei jede der Partitionen einmal die Testmenge, während auf allen anderen trainiert wird; die Fehlerrate und die kumulierte Log-Likelihood Differenz ist dann der Durchschnitt über alle Trainingsbeispiele.

Die zahlreichen Tabellen und Abbildungen mit den Ergebnissen der Tests stehen im Anhang Ergebnisse Generierung Scheinwort-Akronyme auf Seite 25. Darunter befinden sich auch Tabellen mit den zehn besten Akronymen für die Quellwörter *Der tolle rote Wagen* sowie eine Übersicht mit den besten generierten Akronymen aller Scheinwort-Modell-Varianten.

7. Auswertung

Realwort-/Scheinwort-Akronym-Generierung

Zwischen den beiden Arten der Akronymgenerierung gibt es deutliche Unterschiede was die Fehlerrate betrifft. Während bei den Realwort-Akronymen nur runde 15 Prozent der Testbeispiele fehlerhaft generiert wurden, sind es selbst bei der besten Scheinwort-Variante 48 Prozent. Auch die Rang-Abdeckungs-Kurven zeigen die bessere Leistung der Realwort-Suche, denn bei dieser sind schon 90 Prozent der Akronymbeispiele auf den ersten beiden Rängen (Abbildung 5), bei der Scheinwort-Suche sind selbst im besten Fall nur 80 Prozent der Beispiele unter den ersten 100 (Abbildung 8).

Dies mag zum Einen am wesentlich eingeschränkten Suchraum bei der Realwort-Variante liegen, der ja mit einer Liste von echten Wörtern eindeutig eingeschränkt ist. Andererseits ist das N-Gram-Modell in der verwendeten Variante wohl einfach nicht geeignet, die Güte des Wortes zu definieren.

Normierung

Innerhalb der Scheinwort-Akronymgenerierung ist die Normierung des Akronymmodells vorteilhaft. Während bei den nicht normierten Varianten die Fehlerraten zwischen 69 und 76 Prozent liegen, so sind diese bei Normierung in einem Bereich zwischen 48 bis 64 Prozent; auch der Anteil der nicht generierten Akronyme liegt bei den nicht normierten Modellen höher.

Die bessere Leistung der normierten Modelle wird besonders in Abbildung 6 deutlich. Mit Normierung werden im besten Fall doppelt so viele der Testbeispiele richtig generiert, im schlechtesten Fall immer noch zehn Prozentpunkte mehr. Andererseits wird die Zahl der total fehlgeschlagenen Testbeispiele durchweg gar nicht oder nur um wenige Prozentpunkte verringert, siehe dazu Abbildung 7.

Der Grund dafür liegt darin, dass ohne Normierung die Länge des zu bewertenden Akronyms eine wichtige Rolle spielt. Je länger ein Akronym, desto mehr Faktoren gehen in die Bewertungsfunktion ein und desto kleiner wird die Wahrscheinlichkeit. Die Tabellen 12 und 14 zeigen die generierten Akronyme zu dem Wagen-Beispiel unter anderen mit nicht normierten Modell. Unabhängig vom Rest des Modells ist das Akronym *TR* das beste; bei dem 4-Gram-Modell sind drei der ersten fünf Plätze die Einzelbuchstaben *R*, *T* und *D*, die anderen *TR* und *DR*. Von Aussprechbarkeit kann hier keine Rede sein.

Kurzum, die Unabhängigkeit von der Länge des Akronyms durch Normierung scheint äußerst günstig zu sein.

Grad des N-Gram-Modells

Der kleinste getestete N-Gram-Modell Grad ist auch der beste, denn mit steigendem Grad steigt auch in allen Varianten die Fehlerrate und die Rate der nicht generierten Akronyme an. Das 2-Grad-Modell hat auch unabhängig vom Rest des Modells die wenigsten total fehlgeschlagenen Akronymbeispiele – immer runde 20 Prozent.

Der kleinere Grad des N-Gram-Modells bewirkt eine geringere Kontextabhängigkeit bei der Bewertung jedes Buchstabens im Akronym, fixiert damit weniger die Wörter auf denen ursprünglich trainiert wurde, als ein hoher Grad dies tut. Im Wagen-Beispiel für 2-Gram-Modell (Tabelle 13) sieht man, dass dies auch Nachteile hat. Die ersten vier Akronyme *DETRW*, *DTRW*, *DTROW* und *DTRWA* enthalten zwar alle Anfangsbuchstaben, sind also vom Buchstabenzuordnungsmodell her gut, aber nicht aussprechbar. Hier liegt auch der Knackpunkt des Beispiels *Der tolle rote Wagen*, denn die Anfangsbuchstaben der Worte sind Konsonanten. Das 4-Gram-Modell (Tabelle 12) ist hier besser, denn neben den Anfangsbuchstaben sind auch die Vokale der Zweitbuchstaben vertreten, wodurch sieben der zehn besten Akronyme aussprechbar sind. Bei vielen Instanzen der Trainings- und Testbeispiele beginnen jedoch auch Wörter mit Vokalen, vielleicht ein Grund der guten Performanz des 2-Gram-Modells auf der Testmenge.

Unerwarteterweise sinkt die durchschnittliche Log-Likelihood-Differenz mit steigendem Grad. Dies hängt wohl damit zusammen, dass im 2-Gram-Modell in vielen Fällen die erwarteten Akronyme noch generiert werden und schlechtere Plätze in der Rangliste belegen, welche aber bei höherem N-Gram-Grad vielleicht gar nicht generiert wurden. Das ist auch in Abbildung 8 zu beobachten, wo die Abdeckung von Platz zehn zu Platz 100 bei den niedrigeren Graden stark, bei den höheren sehr schwach zunimmt.

Beachtung Wortanfang

Die extra Unterscheidung nach Wortanfang bei der Zählung der N-Gramme führte bei den normierten Testläufen durchgehend zu einer Verschlechterung, bei den nicht normierten dagegen meist zu einer leichten Verbesserung des Ergebnisses. Dabei ist der positive Effekt der Beachtung des Wortanfangs bei nicht normierten Modellen jedoch nicht so groß, wie wenn die Normierung durchgeführt werden würde (Abbildung 6). Unabhängig von der Normierung sind bei 3-, 4- und 5-Gram-Modell eine Zunahme der total fehlgeschlagenen Generierungen um 5 bis 15 Prozentpunkte zu beobachten (Abbildung 7).

Wird der Wortanfang beim N-Gram-Modell beachtet, so gilt für den Anfang des Akronyms, dass die Wahrscheinlichkeit umso höher ist, je mehr Worte beim Training des N-Gram-Modells mit den selben Buchstaben begonnen haben. Beim 2-Gram-Modell betrifft dies nur die ersten beiden Buchstaben, beim 5-Gram-Modell die ersten fünf, womit die Wahrscheinlichkeit sinkt. Dies könnte den starken negativen Effekt der Beachtung des Wortanfangs vor allem bei den hohen N-Gram-Graden erklären.

Beim Wagen-Beispiel hat die Beachtung des Wortanfangs einen positiven Effekt beim 3-Gram-Modell, denn statt *DTROW* ist *DETROW* das beste Akronym, was wesentlich besser aussprechbar ist.

Bereinigung des Wörterbuchs

Die Unterschiede zwischen den Versionen mit bereinigten und nicht bereinigten Wörterbuch auf dem das N-Gram-Modell trainiert wurde, sind erwartungsgemäß minimal. Mal ist die eine Variante besser mal die andere, ein anders Mal keine; aber stets sind die Unterschiede unter einem Prozentpunkt.

Benutzung gemischter Wörterbücher

Die Ergebnisse des Testdurchlaufs mit gemischten Wörterbüchern sind in Tabelle 11 aufgeführt und können mit Tabelle 7 verglichen werden. Die Menge der Testbeispiele ist hier größer, weil auch die deutschen Akronyme benutzt werden. Bei der gemischten Modell-Variante sind die Fehlerraten immer zwei bis drei Prozentpunkte größer, die Rate der nicht generierten Akronyme im 2-Gram-Modell schlechter, sonst besser. Einen richtigen Vorteil bringt das erweiterte Wörterbuch also nicht.

Buchstabenzuordnungsmodell

Hinsichtlich des Buchstabenzuordnungsmodells habe ich die beiden Varianten Position und Position mit Stoppwort getestet. Die Hinzunahme des Stoppwort-Attributs führte zwar fast immer zu einer leichten Verschlechterung der durchschnittlichen Log-Likelihood Differenz, jedoch nie zu einer Verschlechterung der Fehlerrate. Bei der Generierung von Real- und Scheinwort-Akronymen mit Normierung wurde die Fehlerrate in drei von neun Fällen um wenige Zehntel Prozentpunkte besser, blieb sonst aber unverändert. Eine Verbesserung von der Fehlerrate um durchschnittlich 3,6% konnte jedoch bei den nicht normierten Testfällen erreicht werden.

Zusammenfassung

Zusammenfassend lässt sich sagen, dass die Realwort-Akronymgenerierung mit nur 15% Fehlerrate wesentlich besser funktionierte als die Generierung von Scheinwort-Akronymen und bei letzterem ein 2-Gram-Modell mit Normierung ohne Berücksichtigung des Wortanfangs und ohne Stoppwort-Attribut am geeignetsten ist.

Interessant für weitere Untersuchungen wäre die Integration einer Wortsegmentierung für deutsche Wortkomposita und die Interpretierung der Teilwörter als einzelne Worte. Im Englischen sind zusammengesetzte Wörter auseinander geschrieben, womit das vorliegende Akronym-Generierungsmodell auch Anfangsbuchstaben der Teilwörter eher in das Akronym aufnimmt, was im Deutschen nicht der Fall ist. Ein Beispiel für ein deutsches Akronym, in dem die Unterteilung in Teilwörter wichtig wäre, ist *STUPA* – **Studierenden**parlament.

Außerdem könnte man versuchen, semantische Ähnlichkeit des generierten Akronyms zu den Originalwörtern zu modellieren. Denkbar wäre etwa bei der Realwort-Generierung zu messen, zu wie vielen Wörtern der Quellwörter und in welchem Grad das Akronym semantisch verwandt ist. Also wenn etwas aus dem Automobilbereich durch ein Akronym abgekürzt werden soll, könnten dann Akronyme, die etwas mit Autos zu tun haben, eine höhere Wahrscheinlichkeit bekommen.

Anhang A: Ergebnisse Generierung Scheinwort-Akronyme

Die folgenden vier Tabellen 7 bis 10 zeigen die Ergebnisse der Tests auf dem bereinigten englischen Wörterbuch, Tabelle 11 wurde zum Vergleich auf einem gemischten deutschen und englischen bereinigten Wörterbuch erstellt. Im Gegensatz zu den Ergebnistabellen der Realwort-Akronymgenerierung haben diese Tabelle eine zusätzliche Spalte mit der Anzahl und dem Anteil nicht generierter Akronyme.

Die Abbildungen 6 und 7 zeigen die Testergebnisse für die Scheinwort-Akronym-Generierung mit auf dem englischen Wörterbuch trainierten N-Gram-Modellen. In Abbildung 6 ist der Anteil der erfolgreich generierten Akronyme je Modellvariante aufgelistet. Als erfolgreich generiert gilt ein Akronymbeispiel, wenn das entsprechende Alignment die höchste Punktzahl erreicht. Der Anteil der Beispiele, deren Akronyme gar nicht generiert werden konnten, oder die nach ihrer Wahrscheinlichkeit nicht mehr unter den 100 Besten liegen, sind in Abbildung 7 dargestellt.

<i>N-Gram Grad</i>	<i>Attribute</i>	<i>Korrekt generierte Akronyme</i>	<i>Fehlerrate</i>	<i>Durchschnitt. Log-Likelihood Differenz</i>	<i>Anteil nicht generierter Akronyme</i>
2	Position	458 von 875	47,66%	4,34	59 - 6,74%
	Position und Stoppwörter	458 von 875	47,66%	4,4	59 - 6,74%
3	Position	438 von 875	49,94%	4,16	110 - 12,57%
	Position und Stoppwörter	438 von 875	49,94%	4,24	110 - 12,57%
4	Position	382 von 875	56,34%	3,68	259 - 29,60%
	Position und Stoppwörter	387 von 875	55,77%	3,87	260 - 29,71%
5	Position	394 von 875	54,97%	2,88	326 - 37,26%
	Position und Stoppwörter	399 von 875	54,40%	3,13	327 - 37,37%

Tabelle 7: Ergebnis Scheinwort-Akronymgenerierung, normiert

<i>N-Gram Grad</i>	<i>Attribute</i>	<i>Korrekt generierte Akronyme</i>	<i>Fehlerrate</i>	<i>Durchschnitt. Log-Likelihood Differenz</i>	<i>Anteil nicht generierter Akronyme</i>
2	Position	456 von 875	47,89%	4,29	63 - 7,20%
	Position und Stoppwörter	456 von 875	47,89%	4,35	72 - 8,23%
3	Position	407 von 875	53,49%	4,05	176 - 20,11%
	Position und Stoppwörter	407 von 875	53,49%	4,15	177 - 20,23%
4	Position	320 von 875	63,43%	3,58	410 - 46,86%
	Position und Stoppwörter	325 von 875	62,86%	3,84	410 - 46,86%
5	Position	325 von 875	62,86%	2,68	455 - 52,00%
	Position und Stoppwörter	327 von 875	62,63%	2,86	455 - 52,00%

Tabelle 8: Ergebnis Scheinwort-Akronymgenerierung, normiert, Beachtung Wortanfang

<i>N-Gram Grad</i>	<i>Attribute</i>	<i>Korrekt generierte Akronyme</i>	<i>Fehlerrate</i>	<i>Durchschnitt. Log-Likelihood Differenz</i>	<i>Anteil nicht generierter Akronyme</i>
2	Position	221 von 875	74,74%	2,87	133 - 15,20%
	Position und Stoppwörter	268 von 875	69,37%	3,01	131 - 14,97%
3	Position	209 von 875	76,11%	3,01	161 - 18,40%
	Position und Stoppwörter	247 von 875	71,77%	3,21	159 - 18,17%
4	Position	211 von 875	75,89%	2,82	283 - 32,34%
	Position und Stoppwörter	253 von 875	71,09%	3,13	280 - 32,00%
5	Position	215 von 875	75,43%	2,18	334 - 38,17%
	Position und Stoppwörter	256 von 875	70,74%	2,44	335 - 38,29%

Tabelle 9: Ergebnis Scheinwort-Akronymgenerierung, nicht normiert

<i>N-Gram Grad</i>	<i>Attribute</i>	<i>Korrekt generierte Akronyme</i>	<i>Fehlerrate</i>	<i>Durchschnitt. Log-Likelihood Differenz</i>	<i>Anteil nicht generierter Akronyme</i>
2	Position	251 von 875	71,31%	3,21	137 - 15,66%
	Position und Stoppwörter	271 von 875	69,03%	3,19	124 - 14,17%
3	Position	228 von 875	73,94%	3,37	222 - 25,37%
	Position und Stoppwörter	260 von 875	70,29%	3,55	212 - 24,23%
4	Position	220 von 875	74,86%	2,98	422 - 48,23%
	Position und Stoppwörter	236 von 875	73,03%	3,09	417 - 47,66%
5	Position	225 von 875	74,29%	2,31	456 - 52,11%
	Position und Stoppwörter	240 von 875	72,57%	2,38	457 - 52,23%

Tabelle 10: Ergebnis Scheinwort-Akronymgenerierung, nicht normiert, Beachtung Wortanfang

<i>N-Gram Grad</i>	<i>Attribute</i>	<i>Korrekt generierte Akronyme</i>	<i>Fehlerrate</i>	<i>Durchschnitt. Log-Likelihood Differenz</i>	<i>Anteil nicht generierter Akronyme</i>
2	Position	470 von 954	50,73%	4,28	83 - 8,70%
	Position und Stoppwörter	470 von 954	50,73%	4,28	88 - 9,22%
3	Position	446 von 954	53,25%	4,17	109 - 11,43%
	Position und Stoppwörter	446 von 954	53,25%	4,23	109 - 11,43%
4	Position	396 von 954	58,49%	3,9	250 - 26,21%
	Position und Stoppwörter	397 von 954	58,39%	3,97	255 - 26,73%
5	Position	398 von 954	58,28%	3,04	346 - 36,27%
	Position und Stoppwörter	401 von 954	57,97%	3,36	344 - 36,06%

Tabelle 11: Ergebnis Scheinwort-Akronymgenerierung, normiert, bereinigtes deutsch/englisches Wörterbuch

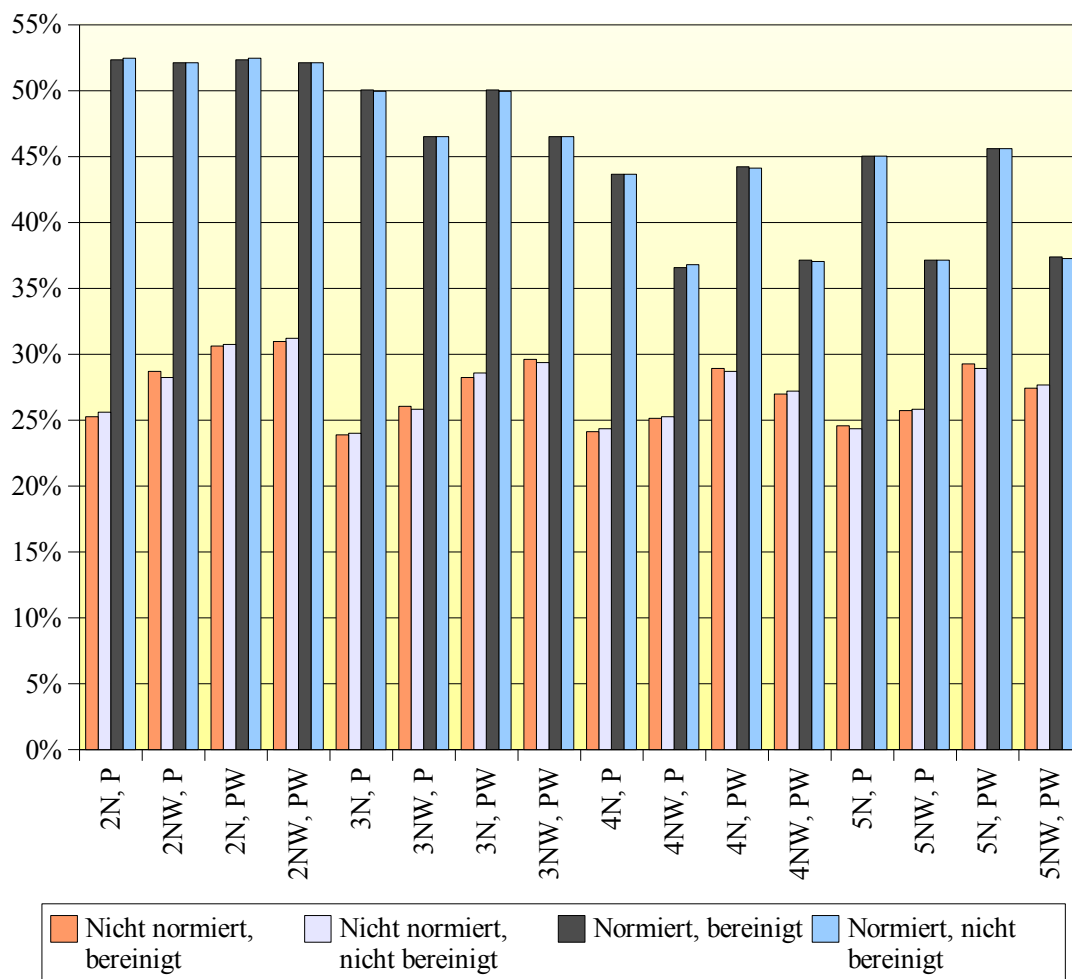


Abbildung 6: Erfolgreich generierte Akronyme in Prozent, englisches Wörterbuch

Legende: 2..5N – N-Gram-Modell Grad 2..5; 2..5NW – N-Gram-Modell mit Berücksichtigung der Wortanfänge Grad 2..5; P – nur Positions-Attribut; PW – Positions- und Stoppwort-Attribut

Ordinate: Anteil der Testbeispiele bei denen das erwartete Alignment die höchste Wahrscheinlichkeit erzielt

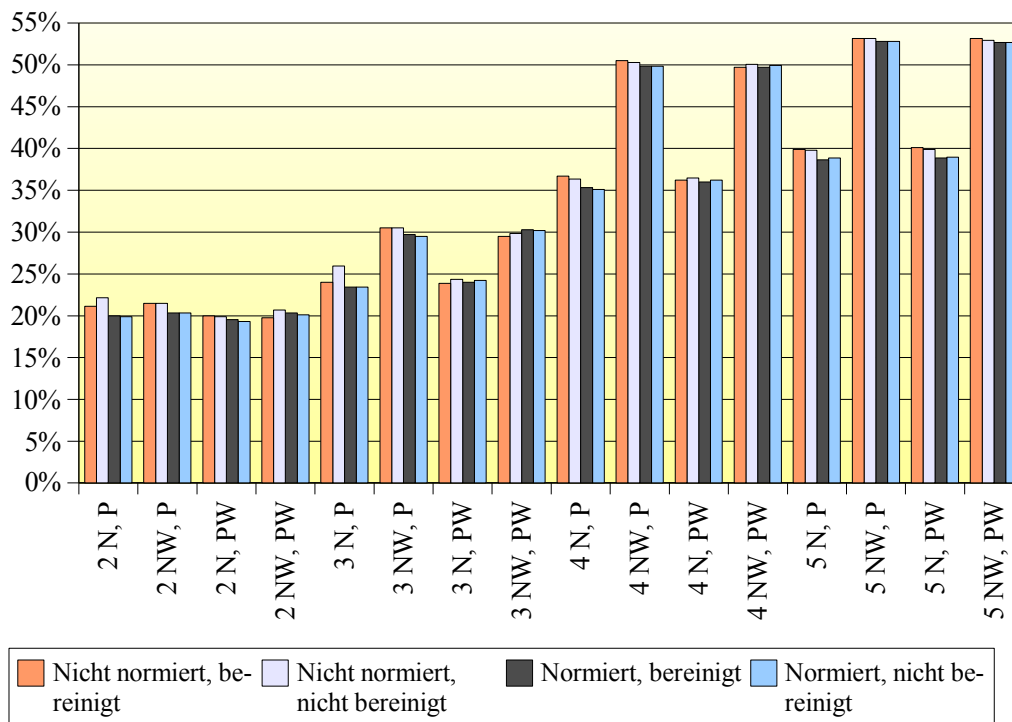


Abbildung 7: Total fehlgeschlagene Akronyme in Prozent, englisches Wörterbuch

Legende: 2..5N – N-Gram-Modell Grad 2..5; 2..5NW – N-Gram-Modell mit Berücksichtigung der Wortanfänge Grad 2..5; P – nur Positions-Attribut; PW – Positions- und Stoppwort-Attribut

Ordinate: Anteil der Testbeispiele bei denen das erwartete Alignment nach Wahrscheinlichkeit nicht unter den ersten 100 ist oder dessen Akronym gar nicht generiert wurde

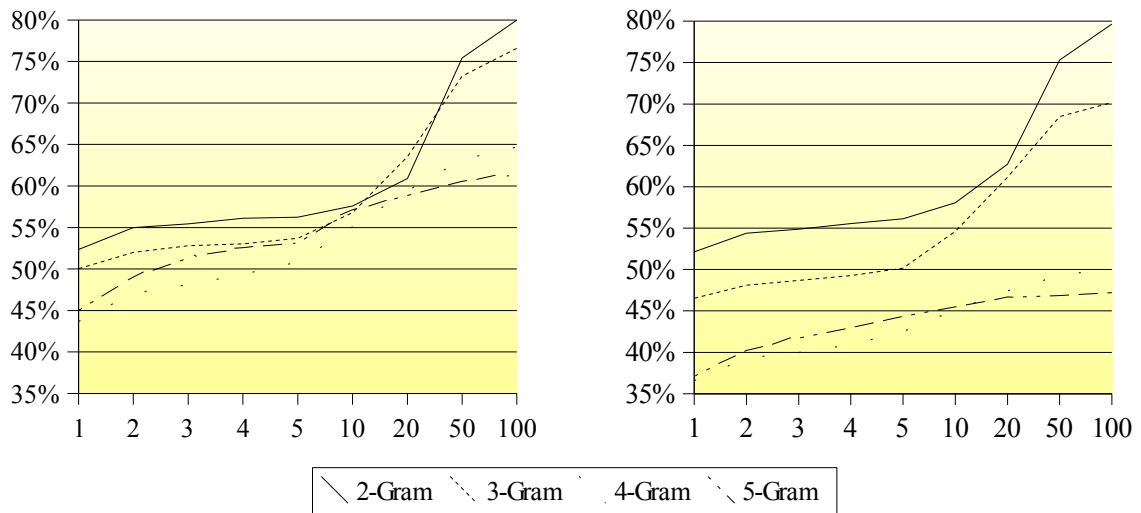


Abbildung 8: Rang-Abdeckungs-Kurve Scheinwort-Generierung nach Grad des N-Gram-Modells

Bereinigtes englisches Wörterbuch, normiert, nur Positions-Attribut; Links: normales N-Gram-Modell mit angegeben Grad je Kurve; Rechts: N-Gram-Modelle mit Beachtung des Wortanfangs

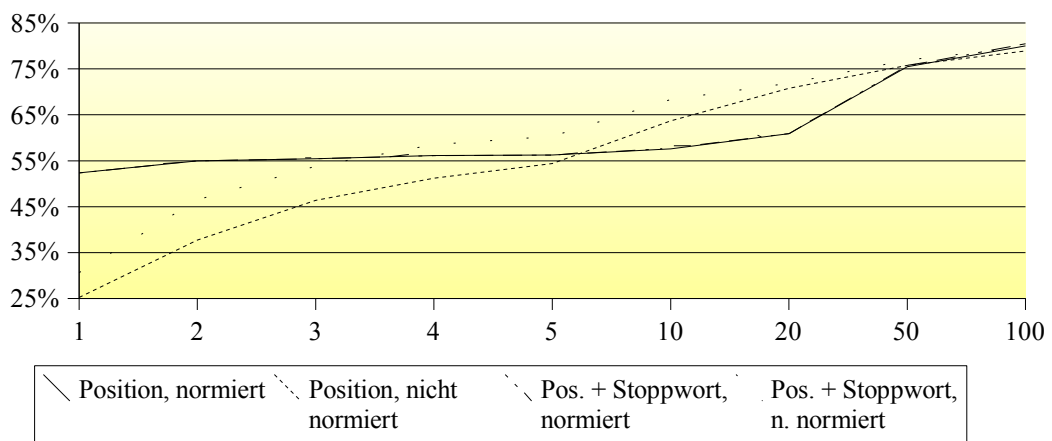


Abbildung 9: Rang-Abdeckungs-Kurven, 2-Gram-Modell, bereinigtes englisches Wörterbuch

4-Gram, Position, gesäubertes englisches Wörterbuch, normiert	4-Gram, Position, gesäubertes englisches Wörterbuch, nicht normiert
DETROW - Der tolle rote Wagen 1	TR - Der tolle rote Wagen 1
DETORW - Der tolle rote Wagen 2	R - Der tolle rote Wagen 2
DETORWA - Der tolle rote Wagen 3	T - Der tolle rote Wagen 3
DETROWA - Der tolle rote Wagen 4	DR - Der tolle rote Wagen 4
DTERW - Der tolle rote Wagen 5	D - Der tolle rote Wagen 5
DETERW - Der tolle rote Wagen 6	DER - Der tolle rote Wagen 6
DERW - Der tolle rote Wagen 7	TRA - Der tolle rote Wagen 7
DETREW - Der tolle rote Wagen 8	TOR - Der tolle rote Wagen 8
DROW - Der tolle rote Wagen 9	TW - Der tolle rote Wagen 9
DETORTW - Der tolle rote Wagen 10	DERW - Der tolle rote Wagen 10

Tabelle 12: Beispiel Scheinwort-Akronymgenerierung mit Quellwörtern *Der tolle rote Wagen*, Vergleich 4-Gram-Modell mit und ohne Normierung

Die schlechte Performanz des nicht normierten Modells wird hier deutlich, denn die ersten fünf Plätze auf der rechten Seite sind weder aussprechbar, noch eine gute Abkürzung, da das Wort „Wagen“ gar nicht vertreten ist.

2-Gram, Position, gesäubertes englisches Wörterbuch, normiert	2Gram, Position + Stoppwort, Beachtung Wortanfang, gesäubertes englisches Wörterbuch, normiert
DETRW - Der tolle rote Wagen 1	DETRW - Der tolle rote Wagen 1
DTRW - Der tolle rote Wagen 2	TRW - Der tolle rote Wagen 2
DTROW - Der tolle rote Wagen 3	DTRW - Der tolle rote Wagen 3
DTRWA - Der tolle rote Wagen 4	DETROW - Der tolle rote Wagen 4
DETROW - Der tolle rote Wagen 5	TROW - Der tolle rote Wagen 5
DTORW - Der tolle rote Wagen 6	DTROW - Der tolle rote Wagen 6
DETRWA - Der tolle rote Wagen 7	DRTRW - Der tolle rote Wagen 7
DRTRW - Der tolle rote Wagen 8	DETRWA - Der tolle rote Wagen 8
DETORW - Der tolle rote Wagen 9	DETORW - Der tolle rote Wagen 9
DTROWA - Der tolle rote Wagen 10	TRWA - Der tolle rote Wagen 10

Tabelle 13: Beispiel Scheinwort-Akronymgenerierung mit Quellwörtern *Der tolle rote Wagen*, Vergleich 2-Gram-Modell ohne und mit Stoppwort-Attribut und Beachtung des Wortanfangs

		<i>Modell</i>		<i>Bestes Alignment</i>		
normiert	bereinigt	2	N	P	DETRW - Der tolle rote W agen	
			PS	DETRW - Der tolle rote W agen		
			NW	P	DETRW - Der tolle rote W agen	
			PS	DETRW - Der tolle rote W agen		
		3	N	P	DTROW - Der tolle rote W agen	
			PS	DTROW - Der tolle rote W agen		
			NW	P	DETROW - Der tolle rote W agen	
			PS	DETROW - Der tolle rote W agen		
		4	N	P	DETROW - Der tolle rote W agen	
			PS	DETROW - Der tolle rote W agen		
			NW	P	DETROW - Der tolle rote W agen	
			PS	TROW - Der tolle rote W agen		
		5	N	P	DERW - Der tolle rote W agen	
			PS	TROW - Der tolle rote W agen		
			NW	P	DROW - Der tolle rote W agen	
			PS	TROW - Der tolle rote W agen		
		nicht bereinigt	2	N	P	DETRW - Der tolle rote W agen
				PS	DETRW - Der tolle rote W agen	
				NW	P	DETRW - Der tolle rote W agen
				PS	DETRW - Der tolle rote W agen	
	3		N	P	DTROW - Der tolle rote W agen	
			PS	DTROW - Der tolle rote W agen		
			NW	P	DETROW - Der tolle rote W agen	
			PS	DETROW - Der tolle rote W agen		
	4		N	P	DETROW - Der tolle rote W agen	
			PS	DETROW - Der tolle rote W agen		
			NW	P	DETROW - Der tolle rote W agen	
			PS	TROW - Der tolle rote W agen		
	5		N	P	DERW - Der tolle rote W agen	
			PS	TROW - Der tolle rote W agen		
			NW	P	DETR - Der tolle rote W agen, DROW - Der tolle rote W agen	
			PS	TROW - Der tolle rote W agen		

		Modell		Bestes Alignment	
nicht normiert	bereinigt	2	N	P	TR - Der tolle rote Wagen
			PS	TR - Der tolle rote Wagen	
		NW	P	TR - Der tolle rote Wagen	
			PS	TR - Der tolle rote Wagen	
		3	N	P	TR - Der tolle rote Wagen
			PS	TR - Der tolle rote Wagen	
		NW	P	TR - Der tolle rote Wagen	
			PS	TR - Der tolle rote Wagen	
		4	N	P	TR - Der tolle rote Wagen
			PS	TR - Der tolle rote Wagen	
		NW	P	TR - Der tolle rote Wagen	
			PS	TR - Der tolle rote Wagen	
	5	N	P	TR - Der tolle rote Wagen	
		PS	TR - Der tolle rote Wagen		
	NW	P	TR - Der tolle rote Wagen		
		PS	TR - Der tolle rote Wagen		
	nicht bereinigt	2	N	P	TR - Der tolle rote Wagen
				PS	TR - Der tolle rote Wagen
			NW	P	TR - Der tolle rote Wagen
				PS	TR - Der tolle rote Wagen
3		N	P	TR - Der tolle rote Wagen	
			PS	TR - Der tolle rote Wagen	
		NW	P	TR - Der tolle rote Wagen	
			PS	TR - Der tolle rote Wagen	
4		N	P	TR - Der tolle rote Wagen	
			PS	TR - Der tolle rote Wagen	
		NW	P	TR - Der tolle rote Wagen	
			PS	TR - Der tolle rote Wagen	
5		N	P	TR - Der tolle rote Wagen	
			PS	TR - Der tolle rote Wagen	
		NW	P	TR - Der tolle rote Wagen	
			PS	TR - Der tolle rote Wagen	

Tabelle 14: Beispiel Scheinwort-Akronymgenerierung mit Quellwörtern Der tolle rote Wagen, Vergleich aller N-Gram-Modell-Variationen mit dem englischen Wörterbuch

Legende: 2..5 Grad des N-Gram-Modells; N – normal, NW – mit Beachtung Wortanfang; P – nur Position-Attribut; PS – Positions- und Stoppwort-Attribut

Anhang B: Verzeichnisse

Literaturverzeichnis

- [Medline05] Medline Fact Sheet; 08.06.2005; <http://www.nlm.nih.gov/pubs/factsheets/medline.html>
- [PUST02] Pustejovsky, James; Castano, José; Cochran, Brent; Kotecki, Maciej; Morrell, Michael; Rumshisky, Anna; *Linguistic Knowledge Extraction from Medline: Automatic Construction of an Acronym Database*; 2002
- [SCH02] Schwartz, Ariel S.; Hearst, Marti A.; *A simple Algorithm for Identifying Abbreviation Definitions in Biomedical Text*; 2002
- [TSU05] Tsuruoka, Yoshimasa; Ananiadou, Sophia; Tsujii, Jun'ichi; *A Machine Learning Approach to Acronym Generation*; 2005
- [MAC03] MacKay, David J.C.; *Information Theory, Inference, and Learning Algorithms*; 2003
- [MAN03] Manning, Christopher D.; Schütze, Hinrich; *Foundations of Statistical Natural Language Processing*; 2003

Abbildungsverzeichnis

Abbildung 1: Noisy-Channel-Modell in der Informationstheorie.....	6
Abbildung 2: Noisy-Channel-Modell in der maschinellen Übersetzung.....	6
Abbildung 3: Zählung für ein 2-Gram-Modell mit Berücksichtigung des Wortanfangs.....	11
Abbildung 4: Beispiel Rang-Abdeckung-Kurve.....	18
Abbildung 5: Rang-Abdeckungs-Kurven Realwort-Akronymgenerierung.....	20
Abbildung 6: Erfolgreich generierte Akronyme in Prozent, englisches Wörterbuch.....	28
Abbildung 7: Total fehlgeschlagene Akronyme in Prozent, englisches Wörterbuch.....	29
Abbildung 8: Rang-Abdeckungs-Kurve Scheinwort-Generierung nach Grad des N-Gram-Modells.....	30
Abbildung 9: Rang-Abdeckungs-Kurven, 2-Gram-Modell, bereinigtes englisches Wörterbuch.....	30

Tabellenverzeichnis

Tabelle 1: Beispiele für Akronyme.....	8
Tabelle 2: Übersicht Symbole.....	9
Tabelle 3: Übersicht Symbole Dekodierung.....	15
Tabelle 4: Ergebnis Suche Realwort-Akronyme, bereinigt.....	20
Tabelle 5: Ergebnis Suche Realwort-Akronyme, nicht bereinigt.....	20
Tabelle 6: Beispiel Realwort-Akronymgenerierung mit Quellwörtern Der tolle rote Wagen, englisches Wörterbuch, nicht bereinigt.....	21
Tabelle 7: Ergebnis Scheinwort-Akronymgenerierung, normiert.....	25
Tabelle 8: Ergebnis Scheinwort-Akronymgenerierung, normiert, Beachtung Wortanfang.....	26
Tabelle 9: Ergebnis Scheinwort-Akronymgenerierung, nicht normiert.....	26
Tabelle 10: Ergebnis Scheinwort-Akronymgenerierung, nicht normiert, Beachtung Wortanfang.....	27
Tabelle 11: Ergebnis Scheinwort-Akronymgenerierung, normiert, bereinigtes deutsch/englisches Wörterbuch.....	27
Tabelle 12: Beispiel Scheinwort-Akronymgenerierung mit Quellwörtern Der tolle rote Wagen, Vergleich 4-Gram-Modell mit und ohne Normierung.....	31
Tabelle 13: Beispiel Scheinwort-Akronymgenerierung mit Quellwörtern Der tolle rote Wagen, Vergleich 2-Gram-Modell ohne und mit Stoppwort-Attribut und Beachtung des Wortanfangs.....	31
Tabelle 14: Beispiel Scheinwort-Akronymgenerierung mit Quellwörtern Der tolle rote Wagen, Vergleich aller N-Gram-Modell-Variationen mit dem englischen Wörterbuch.....	33