Karlsruhe Institute of Technology

Klassenbasierte Sprachmodellierung mit neuronalen Netzen

Bachelorarbeit
von

Thomas Zenkel

am Institut für Anthropomatik
der Fakultät für Informatik

Erstgutachter:	Prof. Dr. Alexander Waibel
Zweitgutachter:	Dr. Sebastian Stüker
Betreuender Mitarbeiter:	Diplom-Informatiker Kevin Kilgour

Bearbeitungszeit: 12. Januar 2015 – 11. Mai 2015

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 11. Mai 2015

Thomas Zenkel

Abstract

In dieser Arbeit werden klassenbasierte Sprachmodelle mit neuronalen Netzen erforscht, in ein Spracherkennungssystem integriert und anhand einer Reihe von Vorträgen evaluiert. Bei der Betrachtung des gesamten Vokabulars einer gesprochenen Sprache in der Ausgabeschicht eines neuronalen Netzes ist es beim heutigen Stand der Technik mit einem Spracherkennungssystem meist nicht möglich, Audioaufnahmen in einer angemessenen Zeit zu erkennen. Deswegen wird zuerst eine Referenzimplementierung getestet, die mit dem neuronalen Netz lediglich eine Teilmenge des Vokabulars betrachtet. Im Vergleich zu einem N-Gramm basierten Sprachmodell kann dabei eine absolute Verbesserung der Wortfehlerrate um 0,8% auf 18,4% beobachtet werden.

Um die Größe der Ausgabeschicht des neuronalen Netzes zu beschränken, werden in den klassenbasierten Sprachmodellen die restlichen Wörter des Vokabulars auf Basis vorher erlernter Merkmalsvektoren in Klassen aufgeteilt, die nun auch von dem neuronalen Netz betrachtet werden. Die Wahrscheinlichkeit dieser Wörter wird dann mithilfe ihrer Klassenzugehörigkeit und einer Unigramm-Wahrscheinlichkeit, die die Häufigkeit des Auftretens eines Wortes in seiner Klasse angibt, bestimmt. Dieses Modell wurde für eine unterschiedliche Anzahl an Klassen und mit verschiedenen Parametern, die für die Interpolation zwischen den Wahrscheinlichkeiten des Sprachmodells des neuronalen Netzes und dem N-Gramm Sprachmodell verantwortlich sind, getestet. Zusätzlich wurde überprüft, ob Größenbeschränkungen für die Klassen eine Verbesserung für das Sprachmodell bewirken. Bei einer Aufteilung der Wörter in 1 000 Klassen konnte mit diesem Ansatz die Wortfehlerrate um zusätzliche 0,1% auf 18,3% verbessert werden.

In einem weiteren Ansatz wird die Unigramm-Wahrscheinlichkeit durch die Zugehörigkeit der Wörter zu Unterklassen ersetzt, welche auch von dem neuronalen Netz betrachtet werden. Dabei wurden Architekturen mit einer unterschiedlichen Anzahl an Ausgabeschichten, unterschiedlich strengen Größenbeschränkungen für die Klassenzuordnungen und verschiedenen Interpolationsparametern evaluiert. Mit diesem Ansatz konnte jedoch keine weitere Verbesserung der Wortfehlerrate festgestellt werden, da sich die Wahrscheinlichkeit der einzeln betrachteten Wörter im Gegensatz zum Referenzmodell veränderte und zu einer höheren Fehlerrate führte. Dagegen kann bei separater Betrachtung der restlichen Wörter eine leichte Verbesserung der Wortfehlerrate gemessen werden.

Aufgrund der hohen Abdeckung der Teilmenge des Vokabulars, die mit dem neuronalen Netz einzeln betrachtet wird, kann die Verbesserung der Wortfehlerrate um 0,1% im Vergleich zu den erzielten Verbesserungen des Referenzmodells bereits als ein ansehnliches Ergebnis bezeichnet werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Immer mehr Anwendungen an den Schnittstellen zwischen Menschen und technischen Geräten hängen aufgrund der zunehmend aktiveren Rolle elektronischer Systeme in unserem Alltagsleben von einem statistischen Sprachmodell ab. Zum Beispiel leisten Sprachmodelle bei der Übersetzung eines Textes in eine andere Sprache, dem Kategorisieren und Transkribieren von Ton- und Videoaufnahmen und bei der Erkennung handgeschriebener Texte einen wichtigen Beitrag.

1.1 Motivation

Ein Ziel eines Sprachmodells ist es dabei Regelmäßigkeiten in der Sprache zu erkennen und so die Wahrscheinlichkeit gegebener Wortkombinationen zu bestimmen. Durch diese zusätzlichen Informationen können dann unter anderem unwahrscheinliche Hypothesen, die durch andere Komponenten einer Anwendung aufgestellt wurden, verworfen werden.

Die Aufgabe der automatischen Spracherkennung ist es, gesprochene Sprache zu analysieren und im Anschluss meistens in einen geschriebenen Text zu überführen. Dabei lässt sich ein dafür verwendetes Spracherkennungssystem in verschiedene Komponenten unterteilen. Wie in Abbildung 1.1 dargestellt, wird das Sprachsignal zuerst vorverarbeitet. Dann wird es mithilfe der Informationen des akustischen Modells und des Sprachmodells dekodiert.

Hierbei wird unter allen möglichen Wortfolgen des Vokabulars W die wahrscheinlichste Wortfolge W^* gegeben eines akustischen Signals A gesucht. Dies lässt sich durch den Satz von Bayes zu der Fundamentalformel der Spracherkennung umformen:

$$\begin{aligned} W^* &= \arg\max_W P(W|A) = \arg\max_W \frac{P(W) \cdot p(A|W)}{p(A)} \\ &= \arg\max_W P(W) \cdot p(A|W) \end{aligned}$$

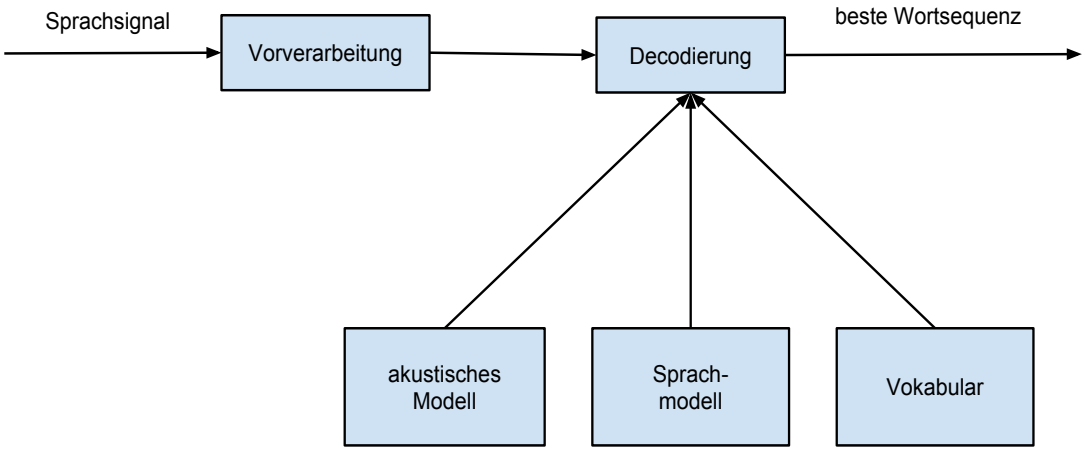


Abbildung 1.1: Blockdiagramm eines Spracherkennungssystems

Wie in [Jeli97] beschrieben, berechnet dabei das Sprachmodell die Wahrscheinlichkeit $P(W)$ und die Wahrscheinlichkeitsdichte $p(A|W)$ wird durch das akustische Modell bestimmt. Die Bedeutung des Sprachmodells zeigt sich vor allem bei Homophonen, also bei Wörtern, die die gleiche Aussprache, aber eine unterschiedliche Bedeutung besitzen. Die Sätze „Er erhielt *mehr* Stimmen in der *Wahl*“ und „Er erhielt *Meer* Stimmen in der *Wal*“ klingen gleich und werden daher im Allgemeinen von dem akustischen Modell als gleich plausibel betrachtet. Mit Hilfe der Grammatik und aufgrund zusätzlicher Informationen aus dem Kontext eines Gespräches ist es für einen Menschen jedoch leicht erkennbar, dass der erste Satz deutlich mehr Sinn ergibt. Dabei sind Homophone nur ein Beispiel, wie das Sprachmodell die Qualität eines Spracherkennungssystems verbessern kann. Auch bei undeutlich ausgesprochenen Wörtern, einem hohen Geräuschpegel und in allen Situationen, in dem das akustische Modell ein Wort nicht eindeutig bestimmen kann, liefert das Sprachmodell wichtige Informationen über die Plausibilität möglicher Wortfolgen.

In der Vergangenheit wurden verschiedene Ansätze erforscht, um robuste Sprachmodelle zu erstellen. Dabei lassen sich die Ansätze grob in deterministische und probabilistische Sprachmodelle unterteilen. Bei einem deterministischen Sprachmodell ist eine bestimmte Wortsequenz erlaubt oder verboten. Zu diesem Zweck kann zum Beispiel mit einer formalen Grammatik definiert werden, welche Wortfolgen in der zu erkennenden Sprache erlaubt sind. Wie in [BaMC05] dargelegt, eignet sich dies besonders bei einem sehr eingeschränkten Dialogsystem, bei dem einem Menschen Fragen gestellt werden, auf die nur wenige Antworten möglich sind. Beispielsweise bei dem automatischen Kauf eines Tickets über das Telefon besitzen formale Grammatiken daher ihre Stärken.

Bei frei gesprochener Sprache haben sich dagegen probabilistische Sprachmodelle als sinnvoller erwiesen, da der Mensch sich nicht immer an die korrekte Grammatik einer Sprache hält. Zudem stellt es eine große Herausforderung dar, alle grammati-

kalischen Regeln einer natürlichen Sprache in einem Sprachmodell zu definieren. Bei probabilistischen Sprachmodellen wird daher einer Wortfolge eine Wahrscheinlichkeit zugewiesen. Diese wird im Allgemeinen anhand eines Trainingstextes geschätzt. Dabei wird in der Regel über die Häufigkeit des Auftretens einer Wortfolge deren Wahrscheinlichkeit geschätzt.

Mit zunehmender Rechenleistung ist es heutzutage aber auch möglich, die Wahrscheinlichkeit einer Wortfolge für eine Teilmenge des Vokabulars einer Sprache in einer annehmbaren Zeit mit einem neuronalem Netz zu erlernen. Dabei stehen neuronale Netze für eine Familie von statistischen Lernalgorithmen, die vom menschlichen Gehirn inspiriert worden sind. Neuronale Netze werden unter anderem erfolgreich in Gebieten wie der Schrifterkennung [WFHS+14], der Bilderkennung [VTBE14] und bei der Vorhersage von Aktienkursen [Tick13] eingesetzt, in denen konventionelle Algorithmen meist nicht in der Lage sind vergleichbare Ergebnisse zu erzielen.

Da die Betrachtung des gesamten Vokabulars bei der Berechnung der Wahrscheinlichkeiten mit einem neuronalen Netz zu rechenaufwendig ist, müssen hierfür andere Methoden verwendet werden. In dieser Arbeit werden die Wörter des Vokabulars daher in Wortklassen unterteilt. Dabei werden verschiedene Verfahren untersucht mit einem neuronalen Netze mithilfe der Wortklassen die Wahrscheinlichkeiten von verschiedenen Wortsequenzen zu bestimmen.

1.2 Gliederung

Diese Arbeit besteht aus insgesamt acht Kapiteln. In Kapitel 2 werden die nötigen Grundlagen der probabilistischen Sprachmodellierung beschrieben. Zuerst werden hierfür N-Gramm Sprachmodelle vorgestellt und danach werden neuronale Netze in der Sprachmodellierung eingeführt. Abschließend werden Maßzahlen zur Evaluation eines probabilistischen Sprachmodells definiert.

In Abschnitt 3.1 des Kapitels 3 werden relevante Arbeiten zu dem in dieser Arbeit behandelten Thema vorgestellt. Dabei werden klassenbasierte N-Gramm Modelle, verschiedene Verfahren zur Sprachmodellierung mit neuronalen Netzen und ein Clusteringalgorithmus, der später zur Bildung von Wortklassen verwendet wird, beschrieben. Danach folgt in Abschnitt 3.2 eine Erörterung der Grenzen von neuronalen Netzen und damit die Motivation des Themas dieser Arbeit.

In dem darauf folgenden Kapitel 4 wird die Architektur des neuronalen Netzes dargestellt. Dabei wird zuerst die grundlegende Architektur eines neuronalen Netzes für die Sprachmodellierung beschrieben. Danach werden die Anpassungen vorgestellt, um klassenbasierte Ansätze mit dem neuronalen Netz zu ermöglichen. Kapitel 5 geht dann auf die Trainingsmethoden des neuronalen Netzes ein.

Die Implementierung der umgesetzten Ansätze wird danach in Kapitel 6 beschrieben. Neben der Implementierung des neuronalen Netzes wird darin auch auf das verwendete Spracherkennungssystem eingegangen. Am Ende dieses Kapitels werden die Verfahren zur Erstellung der Wortklassen detailliert erläutert.

Sowohl die erstellten Wortklassen als auch die implementierten Sprachmodelle werden in Kapitel 7 evaluiert. Das abschließende Kapitel 8 fasst noch einmal die wichtigsten Ergebnisse dieser Arbeit zusammen und schlägt einige weiterführende Möglichkeiten vor, klassenbasierte Sprachmodelle zu entwickeln und einzusetzen.

2. Grundlagen

Dieses Kapitel bietet einen Überblick über grundlegende Verfahren in der statistischen Sprachmodellierung. Dafür wird das N-Gramm Sprachmodell vorgestellt und die damit verbundenen Probleme und ihre Lösungsansätze beschrieben. Danach folgt eine Einführung in die Sprachmodellierung mit neuronalen Netzen und zum Abschluss werden Methoden zur Evaluierung eines gegebenen Sprachmodells vorgestellt.

2.1 Das N-Gramm Sprachmodell

Um die Wahrscheinlichkeit einer Wortsequenz zu schätzen gibt es zwei grundlegende Ansätze. Zum einem kann aus einer Grammatik ein Sprachmodell extrahiert werden. Dieses Prinzip weißt allerdings Wortsequenzen, die in Konversationen durchaus vorkommen können und die allein aufgrund der Grammatik einer Sprache nicht erlaubt sind, eine 0 Wahrscheinlichkeit zu. Des Weiteren lässt sich eine Grammatik nur aufwendig modellieren. Daher werden in modernen Spracherkennungssystemen größtenteils statistische Sprachmodelle verwendet. Wie in [Schu95] beschrieben ist, kann bei einer gegebenen Wortsequenz $w_1, \ldots, w_n$ die Wahrscheinlichkeit $P(w_1, \ldots, w_n)$ der Sequenz folgendermaßen über die bedingten Wahrscheinlichkeiten berechnet werden:

$$P(w_1, \ldots, w_n) = P(w_1) \cdot \prod_{i=2}^{n} P(w_i | w_1 \ldots w_{i-1})$$

Hierbei hängt die Wahrscheinlichkeit eines Wortes $P(w|h)$ von dem Kontext h der ihm vorangegangen Wörter ab. Dadurch steigt bei einem größeren Kontext die Anzahl der zu berechnenden Wahrscheinlichkeiten exponentiell an. Bei einem durchaus üblichen Satz mit 20 Wörtern und einer Vokabulargröße von 300 000 Wörtern müssten 300000^{20} Wahrscheinlichkeiten vorberechnet werden. Da dies nicht realisierbar ist, wird bei einem N-Gramm Sprachmodell die Länge des Kontextes eines Wortes auf N-1 beschränkt. Folglich werden Äquivalenzklassen $c(h)$ gebildet. Damit wird die Annahme getroffen, dass die Wahrscheinlichkeit eines Wortes nur von den N-1 vorangegangen Wörtern abhängt.

Folglich ergibt sich für die Wahrscheinlichkeit einer Wortsequenz:

$$P(w_1, \ldots, w_n) = P(w_1) \cdot \prod_{i=2}^{n} P(w_i | w_{i-N+1}, \ldots, w_{i-1}) \approx \prod_{i=1}^{n} P(w_i | c(h))$$

Ein N-Gramm mit N=1 wird Unigramm genannt, für Bigramme ist N=2 und für Trigramme gilt N=3. N-Gramme mit einem längerem Kontext haben keinen speziellen Namen.

Um die oben genannte Wahrscheinlichkeiten nun abzuschätzen, wird ein Trainingstext gewählt. In diesem Text wird dann gezählt, wie oft ein Wort w in der Äquivalenzklasse seines Kontextes $c(h)$ vorkommt. Im Folgenden bezeichnet # die Anzahl des Auftretens der darauffolgenden Wortsequenz in einem Trainingstext. Damit berechnet sich die gesuchte Wahrscheinlichkeit mit einem N-Gramm Sprachmodell mit der Gleichung:

$$P_{nGram}(w|c(h)) = \frac{\#c(h)w}{\#c(h)}$$

2.2 Smoothing Techniken

Mit dem beschriebenen Ansatz wird die Wahrscheinlichkeit für N-Gramme, die im Trainingstext nicht vorkommen, mit 0 approximiert. Dies lässt sich mit diesem Verfahren nicht vermeiden, da nur ein endlich großer Trainingskorpus gewählt werden kann und dadurch nicht alle Wörter in allen möglichen Kontexten vorkommen werden. Um einem noch nicht gesehenen N-Gramm eine sinnvolle Wahrscheinlichkeit zuzuweisen, werden im Folgenden verschiedene Smoothing Techniken vorgestellt.

Ein einfacher Ansatz ist es davon auszugehen, dass jedes N-Gramm mindestens α-mal im Trainingskorpus vorkommt:

$$P_{Laplace}(w_i|c(h)) = \frac{\#c(h)w_i + \alpha}{\#c(h) + \alpha|V|} \text{ [ChGo96]}$$

Durch diese Verteilung ist trivialerweise die Wahrscheinlichkeit für jedes Wort größer als 0, jedoch wird eine relativ große Wahrscheinlichkeitsmasse hinzugefügt, da pro Wort des Vokabulars α neue Beispiele künstlich erstellt werden. Dies kann vor allem bei einem großen Vokabular die Wahrscheinlichkeitsverteilung stark verfälschen.

Um dies zu vermeiden kann nur ein fester Anteil der Wahrscheinlichkeit $0 < \beta < 1$ umverteilt werden:

$$P(w_i|c(h))_\beta = (1 - \beta)\frac{\#c(h)w_i}{\#c(h)} + \beta\frac{1}{N}$$

In dieser Formel bezeichnet N die Anzahl der Wörter des Vokabulars. Dabei werden aber nun, genau wie bei dem Laplace Smoothing, allen in einem Kontext noch nicht gesehenen Wörtern dieselbe Wahrscheinlichkeit zugeordnet. Um dieses Problem zu vermeiden, können Informationen aus N-Grammen niedrigerer Ordnung verwendet werden.

Dies wird bei der linearen Interpolation, wie in [ChGo96] beschrieben, umgesetzt:

$$P_{interp}(w_i|w_{i-n+1}\dots w_{i-1}) = \\ \lambda P(w_i|w_{i-n+1}\dots w_{i-1}) + (1-\lambda)P_{interp}(w_i|w_{i-n+2}\dots w_{i-1})$$

Dabei bezeichnet λ einen von $w_{i-n+1}\dots w_{i-1}$ abhängigen Skalierungsfaktor. Hier wird jedoch immer auf die weniger akkuraten Informationen der N-Gramme niedrigerer Ordnung zurückgegriffen. Um dies zu vermeiden, greift das *Backoff Smoothing* nur für noch nicht gesehene N-Gramme auf ein Modell niedrigerer Ordnung zurück (vgl. [Schu95]):

$$P_{backoff}(w_i|w_{i-n+1}\dots w_{i-1}) = \begin{cases} P^*(w_{i-n+1}\dots w_i), \text{wenn } \#w_{i-n+1}\dots w_i > 0 \\ \gamma(w_{i-n+1}\dots w_{i-1})P_{backoff}(w_i|w_{i-n-2}\dots w_{i-1}), \text{ sonst} \end{cases}$$

Dabei bezeichnet P^* eine leicht herabgesetzte Verteilung und γ einen Skalierungsfaktor, so dass sich die Wahrscheinlichkeiten zu eins addieren.

2.2.1 Kneser-Ney Smoothing

Eine weitere Smoothing Technik ist das Kneser-Ney Smoothing. Die Idee des in [KnNe95] vorgestellten Verfahrens ist es, die Berechnung der N-Gramme niedrigerer Ordnung zu verbessern. Anstelle die Wahrscheinlichkeit zu betrachten, wie häufig ein N-Gramm vorkommt, wird hier die Anzahl der verschiedenen N-Gramme betrachtet, auf die das betrachtete Wort im Trainingstext folgt:

$$P_{KN}(w_i|w_{i-N+2}\dots w_{i-1}) = \frac{|\{w_{i-N+1}:\#w_{i-N+1}\dots w_i>0\}|}{\sum_{w'}|\{w_{i-N+1}:\#w_{i-N+1}\dots w_{i-1}w'>0\}|}$$

Damit wird *Kneser-Ney Smoothing* dann folgendermaßen definiert:

$$P_{KN}(w_i|w_{i-N+1}\dots w_{i-1}) = P_{abs}(w_i|w_{i-N+1}\dots w_{i-1}) + \alpha \cdot P_{KN}(w_i|w_{i-N+2}\dots w_{i-1})$$

Hierbei beschreibt α wiederum eine vom Kontext abhängige Normalisierungsfunktion und P_{abs} bezeichnet folgende um den konstanten Wert θ herabgesetzte Wahrscheinlichkeit:

$$P_{abs}(w_i|w_{i-N+1}\dots w_{i-1}) = \frac{\max(\#w_{i-N+1}\dots w_i-\theta,0)}{\#w_{i-N+1}\dots w_{i-1}}$$

Der Vorteil dieses Verfahrens verdeutlicht sich am Beispiel von zusammengehörigen Ausdrücken wie dem Städtenamen „San Francisco“. Bei dem Satzanfang „Ich fahre nach“ und den noch nicht gesehenen Bigrammen „nach Karlsruhe“ und „nach Francisco“ wird die Wahrscheinlichkeit des zweiten Bigramms bei Betrachtung der Unigramm-Wahrscheinlichkeiten $P(Karlsruhe)$ und $P(Francisco)$ überschätzt. Die Wahrscheinlichkeit des Auftretens der beiden Bigramme wird gleich hoch geschätzt, wenn die Städtenamen Karlsruhe und San Francisco im Trainingstext vergleichbar oft vorkommen. Da aber das Kneser-Ney Smoothing die Anzahl der N-Gramme betrachtet, auf die ein Wort im Trainingskorpus folgt, wird es den Satz „Ich fahre nach

Karlsruhe.“ mit einer höheren Wahrscheinlichkeit bewerten als den Satz „Ich fahre nach Francisco.“.

In [CCGG98] wurde ein Vergleich von populären Smoothing Verfahren veröffentlicht. Dabei wurden unter anderem Witten-Bell, Katz und Kneser-Ney Smoothing, die unterschiedlich große Trainingsdaten zur Verfügung gestellt bekamen, anhand der Entropie verglichen. Wie auch in Abbildung 2.1 dargestellt, konnte dabei das Kneser-Ney Smoothing unabhängig von der Menge der Trainingsdaten bessere Ergebnisse erzielen als die konkurrierenden Verfahren.

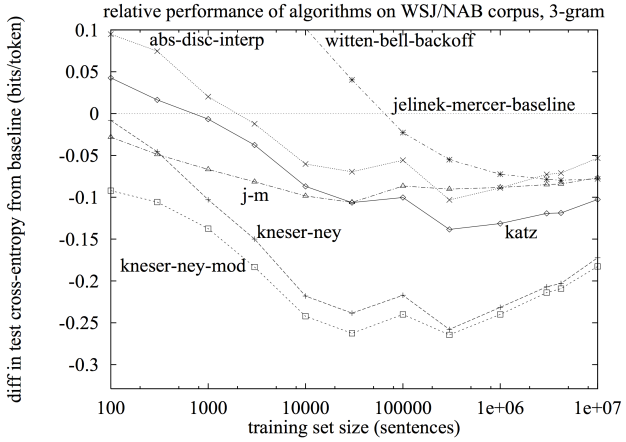


Abbildung 2.1: Vergleich verschiedener Smoothing Verfahren. Quelle: [CCGG98]

2.3 Neuronale Netze in der Sprachmodellierung

Als neuronales Netz wird üblicherweise eine Familie von statistischen Lernalgorithmen, die vom menschlichem Gehirn inspiriert wurden, bezeichnet. Darin bilden viele miteinander verbundene Neuronen ein neuronales Netz. In einem neuronalen Netz fließen über die Neuronen, abhängig von den Gewichten ihrer Verbindungen, Informationen. Diese Gewichte werden anhand eines Trainingskorpus erlernt. Ein einfaches neuronales Netz, gruppiert in eine Eingabeschicht, eine verdeckte Schicht und eine Ausgabeschicht, ist in Abbildung 2.2 dargestellt. Die Kreise stellen dabei Neuronen dar, Pfeile zwischen den Neuronen bedeuten, dass diese Neuronen miteinander verbunden sind. Da bei diesem neuronalen Netz die Neuronen jeder Schicht mit den Neuronen der jeweils nächsten Schicht verbunden sind, wird es als neuronales *feedforward* Netz bezeichnet [Du14].

Mit steigender Rechenleistung wird es heutzutage möglich das Sprachmodell mithilfe eines neuronalen Netzwerkes zu „erlernen“. Bei diesem Ansatz wird zuerst jedem Wort des Vokabulars ein m-dimensionaler Wortvektor $v \in \mathbb{R}^m$ zugeordnet. Danach wird eine Wahrscheinlichkeitsfunktion, die den Wörtern abhängig von ihrem Kontext Wahrscheinlichkeiten zuordnet, mit den Merkmalsvektoren der Wörter des Kontextes ausgedrückt. Sowohl die Merkmalsvektoren als auch die Parameter der Wahrscheinlichkeitsfunktion werden von einem neuronalen Netz gelernt (vgl. [BDVJ03]).

In Abbildung 2.3 wird dieses Konzept dargestellt. Die drei Worte w_{i-3}, w_{i-2} und w_{i-1} werden zuerst auf die bereits zuvor für jedes Wort erlernten Merkmalsvektoren abgebildet. Diese bilden dann die Eingabe für das neuronale Netz. Das neuronale Netz wird so trainiert, dass die Neuronen der Ausgabeschicht die Funktion $P(w_i|w_{i-3}w_{i-2}w_{i-1})$ erlernen.

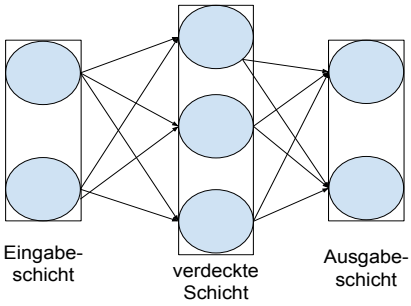


Abbildung 2.2: Einfaches neuronales Netz

Ein Vorteil dieses Vorgehens ist es, dass für ähnliche Wörter üblicherweise auch ähnliche Merkmalsvektoren generiert werden. Bei einer glatten Wahrscheinlichkeitsfunktion wird demselben Wort in einem ähnlichen, aber unterschiedlichen Kontext daher auch eine ähnliche Wahrscheinlichkeit zugeordnet. Zum Beispiel ist zu erwarten, dass $P(Karlsruhe|Ich, fahre, nach) \approx P(Karlsruhe|Ich, fahre, nach)$ gilt. Somit erhöht das Vorkommen eines Wortes in einem Kontext sowohl seine Wahrscheinlichkeit in diesem Kontext als auch die Wahrscheinlichkeit dieses Wortes in einem ähnlichem Kontext.

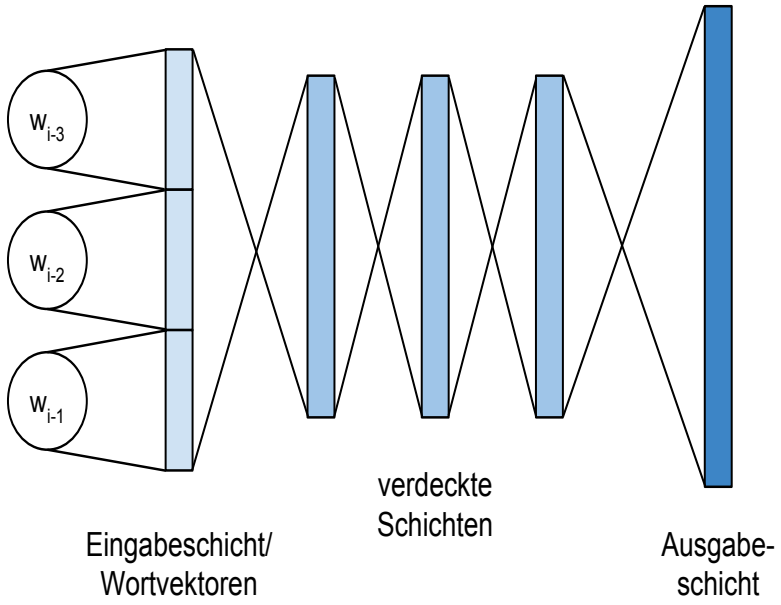


Abbildung 2.3: Neuronales Netz zur Sprachmodellierung

2.4 Evaluation eines Sprachmodells

In diesem Abschnitt werden nun Methoden vorgestellt um die Güte eines Sprachmodells zu bewerten.

2.4.1 Wortfehlerrate

Eine naheliegende Möglichkeit ist es, ein neues Sprachmodell in ein vorhandenes Spracherkennungssystem einzubauen und die Ergebnisse zu vergleichen. Zum Evalu-

ieren eines Spracherkennungssystems bestimmt dieses üblicherweise auf einem aufgenommenen Audiosignal eine Hypothese. Diese Hypothese wird dann mit einer schon vorher von einem Menschen erstellten Referenz verglichen, indem die Wortfehlerrate (englisch: word error rate) WER bestimmt wird. Hierfür wird die niedrigste Anzahl von Vertauschungen V, Löschungen L und Einfügungen E von Worten berechnet um von der Hypothese zur Referenz zu gelangen. Das Ergebnis wird dann durch die Anzahl der Wörter der Referenz N geteilt:

$$\text{WER} = \frac{V+L+E}{N}$$

Ein Nachteil der Wortfehlerrate ist es, dass hierbei nicht nur das Sprachmodell bewertet wird. Wenn zum Beispiel durch das akustische Modell die gesprochene Aufnahme schon gut erkannt wird, werden sich die Wortfehlerrate von zwei verschiedenen Sprachmodellen absolut nicht so stark unterscheiden, wie wenn aufgrund eines schlechten akustischen Modells viele Verbesserungsmöglichkeiten vorhanden wären. Ein weiterer Kritikpunkt an der Wortfehlerrate ist im Deutschen die Erkennung von zusammengesetzten Wörtern. Wenn statt dem Wort „Fußballstadion“ die Wörter „Fuß“ „Ball“ und „Stadion“ erkannt werden, erhöht sich die Wortfehlerrate um $\frac{3}{N}$, da dann zwei Wörter entfernt werden müssen und ein Wort ersetzt wird. Dahingegen erhöht sich beim Verwechseln der Wörter „nein“ und „mein“ die Wortfehlerrate nur um $\frac{1}{N}$, obwohl dieser Fehler den Sinn eines Satzes deutlich stärker verändert.

Der Vorteil dieses Evaluationskriteriums ist es dagegen, dass das Sprachmodell direkt in einem praktischen Anwendungsgebiet getestet wird und dadurch gemessen werden kann, ob ein verbessertes Sprachmodell auch die Qualität eines Spracherkennungssystems verbessert.

2.4.2 Perplexität

Ein anderer Ansatz die Qualität eines Sprachmodells zu evaluieren, ist es die Perplexität des Modells zu berechnen. Die Perplexität beschreibt den mittleren Verzweigungsgrad eines Sprachmodells. Für einen Testdatensatz mit N Wörtern und einem Sprachmodell, das die Wahrscheinlichkeit des Auftretens eines Wortes gegeben seines Kontextes $P(w_i|w_1 \dots w_{i-1})$ bestimmt, ist die Perplexität folgendermaßen definiert:

$$\sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \dots w_{i-1})}} \text{ [Good01]}$$

Ein Sprachmodell, das zehn Wörtern unabhängig von ihrem Kontext die gleiche Wahrscheinlichkeit von 0.1 zuweist, hat somit für jeden aus diesen Wörtern bestehenden Testdatensatz beliebiger Länge die Perplexität 10. Wenn jedoch in dem Testdatensatz zwei Wörter mit der Wahrscheinlichkeit $\frac{1}{3}$ vorkommen und die restlichen acht Wörter mit einer Wahrscheinlichkeit von $\frac{1}{24}$, dann hat der Datensatz eine Perplexität von $\sqrt[3]{3 \cdot 3 \cdot 24} = 6$. Ein Sprachmodell, welches die tatsächliche Wahrscheinlichkeitsverteilung der Wörter eines Textes besser beschreibt, weißt ihm somit auch eine geringere Perplexität zu.

Die Entropie ist über den binären Logarithmus der Perplexität definiert. Dieses Maß beschreibt die durchschnittliche Anzahl an Bits pro Wort, die nötig ist, um einen Text, der genau der Wahrscheinlichkeitsverteilung $P(w_i|w_1 \dots w_{i-1})$ des Sprachmodells entspricht, optimal binär zu kodieren.

3. Verwandte Arbeiten und Problemerörterung

Im ersten Abschnitt dieses Kapitels werden verwandte Arbeiten vorgestellt. Dabei wird zuerst ein klassenbasiertes N-Gramm Sprachmodell beschrieben, welches einen der ersten Ansätze darstellt, Wortklassen in der Sprachmodellierung zu benutzen. Danach werden verschiedene Ansätze zur Sprachmodellierung mit neuronalen Netzen vorgestellt. Darunter finden sich unter anderem rekurrente neuronale Netze und ein neuronales Netz mit mehreren Ausgabeschichten, welches die Zugehörigkeit eines Wortes zu verschiedenen Klassen betrachtet. Abschließend wird noch ein Clusteringverfahren, das später in dieser Arbeit zur Bildung von Wortklassen mit Größenbeschränkungen verwendet wird, beschrieben .

Im zweiten Teil dieses Kapitels wird dann erörtert, warum es sinnvoll ist, ein klassenbasiertes Verfahren bei der Sprachmodellierung mit neuronalen Netzen zu verwenden.

3.1 Verwandte Arbeiten

3.1.1 Klassenbasierte N-Gramm Modelle

Ein Konzept für klassenbasierte N-Gramm Modelle ist es, anstelle der Wahrscheinlichkeiten eines Wortes gegeben seines Kontextes $P(w_i|w_{i-N+1} \dots w_{i-1})$ die Wahrscheinlichkeit des Wortes abhängig von den Wortklassen seines Kontextes zu bestimmen. Dafür wird eine Funktion $c(w)$ benötigt, die einem Wort w eine Klasse zuordnet. Die Wahrscheinlichkeit eines N-Gramms wird dann wie folgt bezeichnet:

$$P(w_i|c(w_{i-N+1}) \dots c(w_{i-1}))$$

Dieses Verfahren hat den Vorteil, dass deutlich weniger Wahrscheinlichkeiten für das Sprachmodell geschätzt werden müssen. Zudem tritt ein Wort im Kontext der Klassen logischerweise häufiger auf als im Kontext der einzelnen Wörter dieser Klassen. Dadurch gibt es in einem Trainingstext deutlich mehr Beispiele zum Schätzen der N-Gramm Wahrscheinlichkeiten.

Friday	Monday	Thursday	weekend	Sunday	Sundays	weekends
great	big	vast	sudden	mere	gigantic	lifelong
head	body	eyes	voice	arm	eye	mouth

Tabelle 3.1: Ausgewählte Wörter der in [BdMP+92] erstellten Klassen

Der beschriebene Ansatz wurde in [BdMP+92] erforscht. Zum Finden von geeigneten Wortklassen wurde die Transinformation zwischen den Klassen maximiert. Die Transinformation I gibt die Stärke des statistischen Zusammenhangs zweier Zufallsgrößen an. Für zwei Klassen c_1 und c_2 ist sie wie folgt definiert:

$$I(c_1, c_2) = \sum_{c_1,c_2} P(c_1 c_2) \log \frac{P(c_2|c_1)}{P(c_2)}$$

Der Gedanke dieses Ansatzes ist, dass die Transinformation zwischen Klassen sich verringert, wenn Wörter zu einer Klasse hinzugefügt werden, die nicht in diese Klasse passen. Zum Beispiel sollte die Transinformation einer Klasse $c_1 = \{lerne, studiere\}$ und der Klasse $c_2 = \{Informatik, Mathematik\}$ hoch sein, da Wörter der Klasse c_2 häufig nach Wörtern der Klasse c_1 vorkommen. Bei Hinzunahme des Wortes *schwimmen* zur Klasse c_1 sinkt die Transinformation jedoch.

Der in [BdMP+92] beschriebene Algorithmus zur Bildung von Wortklassen ordnet den C häufigsten Wörtern jeweils eine eigene Klasse zu. Dann wird nacheinander für das jeweils nächst häufigste Wort eine neue Klasse erstellt und die zwei Klassen vereinigt, für die die durchschnittliche Transinformation der daraus resultierenden Klassenzuordnung am höchsten ist. Mit diesem Verfahren wurden 260 000 Wörter in 1 000 Klassen unterteilt. In Tabelle 3.1 sind Wörter einiger ausgewählter Klassen dargestellt.

Mit einem aus diesen Klassen erstellten, interpolierten 3-Gramm Sprachmodell wurde die Perplexität des Brown Korpus mit 271 bestimmt. Dabei bezeichnet der Brown Korpus einen aus einer Million Wörtern bestehenden englischsprachigen Text, der im Jahr 1961 mit dem Ziel zusammengestellt wurde, die damals verwendete gebräuchliche Sprache möglichst gut zu repräsentieren. Mit dem aus dem Trainingstext, den das klassenbasierte Sprachmodell auch verwendet hatte, erstellten, wortbasierten N-Gramm Modell wurde eine Perplexität von 244 gemessen. Jedoch konnte mit einem aus beiden Ansätzen interpolierten Modell eine Perplexität von 236 erreicht werden.

3.1.2 Neuronale Netze in der Sprachmodellierung

Neuronale Netze wurden bereits 2003 in [BDVJ03] in der Sprachmodellierung eingesetzt. Der grundlegende Ansatz zur Verwendung eines neuronalen Netzes in der Sprachmodellierung ist bereits in Kapitel 2 beschrieben. Dagegen wurde in [BDVJ03] nur eine verdeckte Schicht eingesetzt und das neuronale Netz beschränkte sich auf ein Vokabular von 16 000 Wörtern. Mit einem Trainingstext von 800 000 Wörtern konnte damit auf dem Brown Corpus eine Perplexität von 252 gemessen werden. Dafür wurde das neuronale Netz mit einem Kontext von vier Wörtern mit einem Trigramm Modell interpoliert. Zum Vergleich wurde auf diesem Korpus mit einem 5-Gramm Sprachmodell mit Kneser-Ney Smoothing eine höhere Perplexität von 321 gemessen.

In [Schw07] wurde ein Sprachmodell mit einem neuronalen Netz dann auch in der Sprachtranskribierung eingesetzt. Dabei wurde in einem Spracherkennungssystem zuerst ein N-Gramm Sprachmodell verwendet, um einen Wortgraph mit den wahrscheinlichsten Wortfolgen zu erstellen. Dieser Wortgraph wurde danach mit dem Sprachmodell des neuronalen Netzes neu bewertet.

Bei der Implementierung des neuronalen Netzes wurden dabei einige Optimierungen vorgenommen, um die Dekodierung in einer angemessenen Zeit durchzuführen. Zum Beispiel wurden Wahrscheinlichkeitsanfragen für denselben Kontext für unterschiedliche Wörter zusammengefasst. Außerdem berechnet das neuronale Netz nur die Wahrscheinlichkeiten für eine Teilmenge des gesamten Vokabulars. Die Trainingsdaten für das neuronale Netz stammten dabei aus verschiedenen Ressourcen. Unter anderem wurden Daten aus dem Internet, Zeitungstexte und Transkriptionen aus akustischen Trainingsdaten verwendet. Beim Transkribieren von englischsprachigen Nachrichten konnte die Wortfehlerrate konstant um 0,2% gesenkt werden. Gleichzeitig konnte die Perplexität auf den Testdaten von 148 auf 130 vermindert werden.

3.1.3 Ein klassenbasierter Ansatz mit neuronalen Netzen

Da es aus Effizienzgründen nicht machbar ist, das gesamte Vokabular einer Sprache in der Ausgabeschicht eines neuronalen Netzes zu betrachten, wird in der Ausgabeschicht meist nur eine Shortlist bestehend aus den gebräuchlichsten Wörtern der Sprache verwendet. In [LOAG+13] wird der Ansatz Wortklassen aus dem Vokabular zu bilden und diese Klassen in dem Sprachmodell zu verwenden, wieder aufgegriffen. Indem die nicht in der Shortlist enthaltenen Wörter hierarchisch in Klassen aufgeteilt werden, erlaubt es dieser Ansatz, auch bei einem großen Vokabular alle Wörter in der Ausgabeschicht des neuronalen Netzes zu betrachten.

Die Wortklassen wurden in [LOAG+13] nach dem Lernen der Merkmalsvektoren der Wörter gebildet. Auf den Vektoren wurde rekursiv der K-Means Algorithmus ausgeführt. Hierbei wurde eine Shortlist mit 12 000 Wörtern verwendet und in einem ersten Schritt die restlichen Wörter auf 4 000 Klassen aufgeteilt. Solange eine Klasse noch mehr als 1 000 Wörter besaß, wurde sie rekursiv in mehrere Klassen aufgeteilt. Dabei wurde eine Klasse mit W Wörtern jeweils in $\sqrt{W}+1$ Unterklassen untergliedert. Sei nun D die Anzahl der Klassen und Unterklassen, in die ein Wort unterteilt wurde und $c_k(w)$, die Zuordnung des Wortes w zu seiner k-ten Klasse, dann ist die Wahrscheinlichkeit des Wortes w gegeben seines Kontextes h definiert über:

$$P(w|h) = P(c_1|h) \prod_{d=2}^{D} P(c_d|h, c_1 \dots c_{d-1})$$

Wie in Abbildung 3.1 dargestellt, gibt es nun mehrere Ausgabeschichten. Eine Ausgabeschicht berechnet die Wahrscheinlichkeiten der Wörter in der Shortlist und die Wahrscheinlichkeit der Oberklasse der restlichen Wörter. Für jede Klasse gibt es nun eine weitere Ausgabeschicht, die für die Wahrscheinlichkeit der in der Hierarchie darunterliegenden Klassen zuständig ist. Somit werden bei diesem Ansatz mehr als 4 000 verschiedene Ausgabeschichten benötigt, da für jede Klasse noch mindestens eine eigene Ausgabeschicht nötig ist, um die Wahrscheinlichkeit des Wortes innerhalb seiner Klasse zu approximieren. Aufgrund dieser speziellen Beschaffenheit

des neuronalen Netzes wird die Ausgabeschicht dabei als strukturierte Ausgabeschicht bezeichnet. Deshalb wird bei dieser Arbeit auch von einem SOUL (engl. für Structured OUtput Layer) neuronalen Netz gesprochen.

Mit diesem Modell konnte die Wortfehlerrate bei einem Kontext von drei Wörtern im Vergleich zu einem neuronalen Netz, das in der Ausgabeschicht nur Wörter der Shortlist enthält, absolut um bis zu 0,2 Prozent verbessert werden. Auch bei interpolierten 6-Gramm Modellen wurden durch dieses Verfahren Verbesserungen von 0,2 Prozent erzielt. Gemessen wurden diese Werte mit einem Testdatensatz auf Arabisch und auf Mandarin, wobei für beide Sprachen mit dem neuen Ansatz die genannten Verbesserungen erzielt werden konnten.

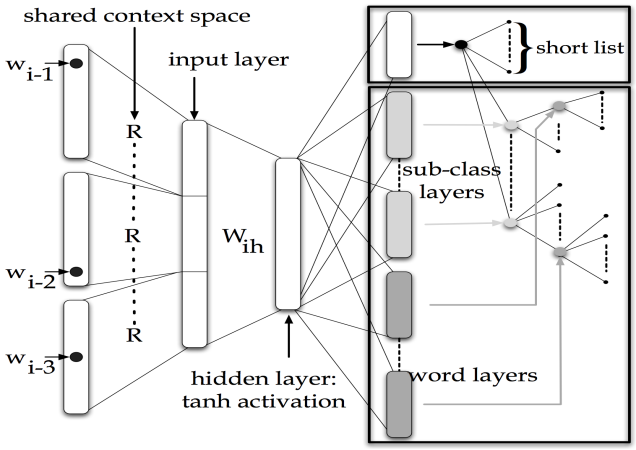


Abbildung 3.1: Architekur des neuronalen Netzes mit einer strukturierten Ausgabeschicht. Quelle: [LOAG+13]

In [Le12] wurde dieser Ansatz zusätzlich mit mehreren verdeckten Schichten getestet. Damit das neuronale Netz dabei weiterhin in einer angemessenen Zeit zu trainieren ist, wurde dafür die Größe der Shortlist auf 2 000 Wörter reduziert. Außerdem wurden die Wörtern in dem ersten Clustering Schritt nur auf 2 000 Klassen aufgeteilt. Durch das Hinzufügen einer zusätzlichen verdeckten Schicht konnte dabei zwar die Perplexität relativ um bis zu 5% gesenkt werden, bei der Spracherkennung konnte aber keine niedrigere Fehlerrate festgestellt werden.

3.1.4 Rekurrente neuronale Netze

Eine Einschränkung der N-Gramm Modelle und der bisher vorgestellen Verfahren zur Sprachmodellierung mit neuronalen Netzen ist die fest gewählte Kontextlänge. Mit rekurrenten neuronalen Netzen ist es jedoch möglich, auch einen Kontext variabler Länge zu betrachten. Dieses Konzept wird unter anderem in [MKBC+10] angewendet. Wie in Abbildung 3.2 dargestellt, besteht dabei das neuronale Netz aus einer Eingabeschicht und einer Ausgabeschicht. Dazwischen befinden sich zusätzlich zwei gegenseitig miteinander verbundene Kontextschichten. Die zyklische Verbindung zwischen den beiden Schichten ermöglicht es, dass nicht mehr nur die Eingaben des aktuellen Zeitschritts mitberücksichtigt werden, sondern ein beliebig langer Kontext betrachtet werden kann. Für das Training des Netzes wird dabei eine auf das Modell angepasste Variation des Backpropagation Algorithmus angewandt.

Getestet wurde dieses neuronale Netz mit Kontextschichten bestehend aus 30 bis 500 Neuronen und einem 5-Gramm Sprachmodell mit Kneser-Ney Smoothing als

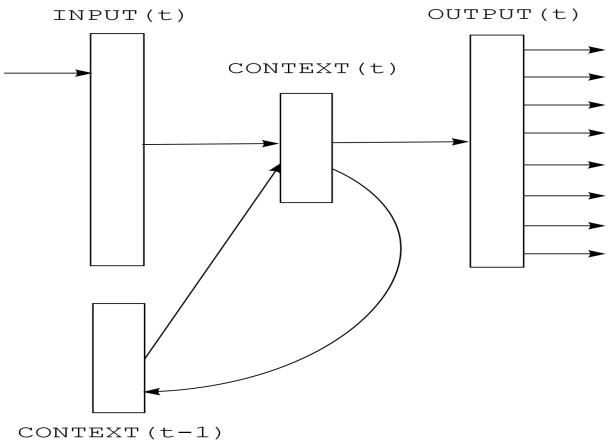


Abbildung 3.2: Einfaches rekurrentes neuronales Netz. Quelle: [MKBC+10]

Backoff Modell. Zwischen den zwei Sprachmodellen wurde linear interpoliert, wobei das neuronale Netz einen Interpolationsparameter von 0,75 zugewiesen bekam. Mit diesem Aufbau konnte die Wortfehlerrate im Vergleich zu dem 5-Gramm Sprachmodell von 13,5% auf bis zu 11,7% und die Perplexität von 221 auf 156 reduziert werden. Für das Training wurde bei diesen Ergebnissen ein aus 6,4 Millionen Wörtern bestehender Korpus verwendet.

Zusätzlich wurde noch der Ansatz evaluiert, dass das rekurrente Netz auch während der Testphase trainiert wird. Dabei wurde eine feste Lernrate verwendet. Damit erhöht sich für Worte, die in den Testdaten häufiger vorkommen und in den Trainingsdaten nicht enthalten waren, die erlernte Wahrscheinlichkeit. Mit diesem dynamischen Ansatz wurde auf dem Wall Street Korpus eine weitere Verbesserung der Wortfehlerrate auf 11,1% und der Perplexität auf 121 erreicht.

3.1.5 Clusteringverfahren mit Größenbeschränkungen

In vielen praktischen Anwendungsfällen ist es sinnvoll, Cluster von ungefähr gleicher Größe zu bilden. In einer Marketingkampagne sollen beispielsweise für verschiedene Kundengruppen unterschiedliche Strategien benutzt werden. Hier sollten die Kundengruppen eine Mindestgröße haben, um für den Umsatz eines Unternehmens relevant zu sein. Um eine Mindestgröße der einzelnen Cluster zu garantieren, wird in [BaGh] ein von dem konkreten Clusteringverfahren unabhängiges Rahmenwerk vorgestellt, welches in drei Schritte unterteilt werden kann:

In einem ersten Schritt wird eine repräsentative Teilmenge der Daten bestimmt. Hierfür wird gezeigt, dass es im Allgemeinen bei zufälliger Wahl der repräsentativen Teilmenge kein Problem darstellt, mindestens eine bestimmte Menge an Datenpunkten jedes Clusters der optimalen Lösung zu wählen. So lässt sich bei 10 Clusterzentren und einer zufällig gewählten Teilmenge mit 1240 Datenpunkten schon mit 99,9 prozentiger Sicherheit sagen, dass mindestens 50 Datenpunkte aus jedem Cluster der optimalen Lösung in der Teilmenge enthalten sind.

Danach werden die Datenpunkte der Teilmenge mit einem Algorithmus in die verschiedenen Cluster aufgeteilt. Der dafür verwendete Algorithmus muss in der Lage sein, die Datenpunkte in k Cluster aufzuteilen und einen Repräsentanten für jedes dieser Cluster zu bestimmen.

Im nächsten Schritt werden nun alle Cluster bis zu der gegebenen Mindestgröße befüllt. Hierbei gilt für jeden Punkt, der einem Cluster zugewiesen wird, eine der folgenden zwei Bedingungen. Entweder wurde der Punkt dem Cluster mit dem ihm ähnlichsten Repräsentanten zugewiesen, oder alle Repräsentanten der Cluster, die dem Punkt ähnlicher sind als der Repräsentant des ihm zugewiesenen Clusters, enthalten schon die Mindestanzahl an Punkten. Nach diesen Zuordnungen haben alle Cluster ihre Mindestgröße erreicht und jeder noch nicht zugeordnete Punkt wird dem ihm ähnlichsten Cluster zugeordnet.

In der letzten Phase wird die Zuordnung der Punkte zu den Clustern noch durch individuelle Neuzuordnungen und Vertauschungen von Punkten verbessert. Wenn ein Cluster näher an einem Punkt liegt als sein aktuelles Cluster und das aktuelle Cluster mehr Punkte als die geforderte Mindestanzahl enthält, wird der Punkt dem ihm näheren Cluster zugeordnet. Dann werden, falls es die Mindestgröße der Cluster nicht verletzt und es zu einer besseren Zuordnung führt, gleichzeitig mehrere Punkte zu anderen Clustern zugeordnet. Diese zwei Schritte werden so oft wiederholt, bis keine erlaubte Neuzuordnung mehr zu einer signifikanten Verbesserung führt.

Wenn der K-Means Algorithmus bei obigen Vorgehen verwendet wird und eine konstante Anzahl an Iterationen sowohl bei dem K-Means Algorithmus als auch bei den Neuzuordnungen festgelegt wird, liegt die Komplexität dieses Verfahrens bei $\mathcal{O}(kN \log N)$. Hierbei bezeichnet k die Anzahl der Cluster und N die Anzahl der aufzuteilenden Punkte.

Das Verfahren wurde an verschiedenen Datensätzen mit bis zu 128 000 Datenpunkten und 46 000 Dimensionen getestet. Als Cluster Algorithmus wurde hauptsächlich der K-Means Algorithmus mit der Kosinus-Ähnlichkeit als Ähnlichkeitsmaß verwendet. Gemessen an der mittleren Transinformation zwischen den Clusterzuweisungen und den davor bestimmten Klassenzugehörigkeiten der einzelnen Daten erzielte das vorgeschlagene Verfahren im Vergleich mit den vom K-Means Algorithmus ohne Größenbeschränkungen erzeugten Klassenzuordnung vergleichbare Ergebnisse. Auf einigen Daten konnten sogar bessere Ergebnisse erzielt werden, da durch die Größenbeschränkung lokale Minima vermieden wurden, die der K-Means Algorithmus aufgrund einer schlechten Initialisierung gefunden hatte.

3.2 Problemerörterung

Wie in den gerade aufgeführten Arbeiten zu sehen ist, wird mit neuronalen Netzen häufig nur ein kleiner Teil des Vokabulars betrachtet. Dies hat mehrere Gründe. Zum einen steigt mit der Anzahl der Wörter des Ausgabevokabulars auch die Anzahl der während der Trainingsphase des Netzes zu schätzenden Parameter. Die Anzahl der Parameter berechnet sich bei n verdeckten Schichten aus:

$$I \cdot H_1 + \sum_{i=1}^{n-1} H_i \cdot H_{i+1} + H_n \cdot O$$

Dabei wird die Anzahl der Neuronen der Eingabeschicht mit I, der i-ten verdeckten Schicht mit H_i und der Ausgabeschicht mit O bezeichnet. Bei gängigen Werten wie $O = 600$ und $H_i = 600$ und $n = 3$ wird die Anzahl der zu schätzenden Parameter durch O dominiert, also durch die Größe des Ausgabevokabulars. Wenn das gesamte Vokabular, das im Deutschen in [Schu95] auf circa 300 000 Wörter geschätzt wird,

O	Zeit in ms
600	2
10 000	43
20 000	113
100 000	570
300 000	1 770
500 000	3 030

Tabelle 3.2: Gemessene Zeiten der Multiplikation eines 600 dimensionalen Vektors mit einer Matrix der Größe 600xO auf einem Quad-Core AMD Opteron Prozessor mit 1,1 GHz

mit dem neuronalen Netz betrachtet wird, müssen daher also $600^2 \cdot 3 + 600 \cdot 300000 > 180$ Millionen Parameter betrachtet werden. Wenn dahingegen nur eine Teilmenge des Vokabulars von 20 000 Wörtern in der Ausgabeschicht betrachtet wird, sind nur noch knapp 13 Millionen Parameter zu schätzen.

Zudem wird während des Erkennens einer Sprachsequenz häufig die Wahrscheinlichkeit für das nächste Wort benötigt. Diese wird für die Erkennung einer Sprachsequenz von 131 Sekunden mehr als 900 000 mal erfragt. Auch wenn 40 000 Wahrscheinlichkeiten mit demselben Kontext in einem Cache zwischengespeichert werden, werden immer noch über 45 000 Anfragen an das neuronale Netz benötigt. Für jede Anfrage müssen dabei die folgenden Matrixmultiplikationen durchgeführt werden, wobei nun I für den Eingabevektor des neuronalen Netzes steht, H_i für die Matrix der Parameter zwischen der verdeckten Schicht i und $i+1$ und O für die Matrix der Parameter zwischen der letzten verdeckten Schicht und der Ausgabeschicht:

$$I \cdot (\prod_{i=1}^{n-1} H_i) \cdot O$$

Die Berechnung der letzten Matrixmultiplikation dieser Gleichung benötigt dabei aufgrund der deutlich größeren Dimensionen der Matrix O die meiste Zeit. Bei einem Ausgabevokabular mit $|S|$ Wörtern wird hier ein Vektor mit $|H|$ Dimensionen mit einer Matrix der Größe $|H| \times |S|$ multipliziert. In Tabelle 3.2 sind dabei die Zeiten der Berechnung dieser Matrixmultiplikation für verschieden Größen des Ausgabevokabulars aufgeführt. Bei einem Ausgabevokabular von 600 Wörtern wurde für diese Multiplikation eine Zeit von 2 ms gemessen. Bei den zur Zeit gebräuchlichen Größen des Ausgabevokabulars des neuronalen Netzes von 20 000 Wörtern benötigt die Matrixmultiplikation 113 ms. Bei der Betrachtung des gesamten Vokabulars würde die Berechnungszeit bei einer Vokabulargröße von 300 000 Wörtern jedoch schon auf 1,7 Sekunden ansteigen. Für die Berechnung von 45 000 Wahrscheinlichkeiten, die für die Erkennung einer Sprachsequenz von 131 Sekunden benötigt werden, würden die dafür benötigten Matrixmultiplikationen allein schon mehr als $45000 \cdot 1.7s > 21h$ in Anspruch nehmen.

Zudem ist die Abdeckung der am häufigsten benutzen Wörter, die normalerweise in der Ausgabeschicht betrachtet werden, in einer Sprache bereits relativ hoch. Bei einem deutschen Trainingstext von 34 Millionen Wörtern wurde mit einem Vokabular

aus den 18 400 häufigsten Wörtern ohne Berücksichtigung der Groß- und Kleinschreibung eine Abdeckung von 91.28% erreicht. 8,72% der Wörter sind demnach nicht in diesem Vokabular enthalten. Dieser Wert wird auch als OOV-Wert (englisch: Out Of Vocabulary) bezeichnet. Auch in Mandarin und Arabisch wurden mit den häufigsten 12 000 Wörtern, wie in [LOAG+13] beschrieben wurde, eine Abdeckung von 90% bzw. 95% erzielt. In [Bazz02] wurde mit einem Vokabular von 25 000 Wörtern eine Abdeckung von 97,3% in einem englischen Text gemessen.

Der größere Unterschied der Abdeckung im Deutschen und im Englischen lässt sich vor allem durch die Unterschiede der Sprachen erklären. Während sich im Englischen die Wörter bei einem unterschiedlichen Kasus und Genus nur selten unterscheiden, verändern sich die Wörter dagegen im Deutschen häufiger. Beispiele hierfür sind die Wörter „Lehrer“ und „teacher“. In der femininen Form verändert sich das Wort im Englischen nicht, wobei im Deutschen dann von der „Lehrerin“ gesprochen wird. Auch bei den verschiedenen Fällen sind leicht Beispiele zu finden, wie unter anderem das Anhängen eines „s“ im Genitiv (der Lehrer, des Lehrers), das im Englischen keine Entsprechung findet. Außerdem erhöhen die zusammengesetzten Wörter im Deutschen, bei denen aus mehreren Wörtern ein neues Wort gebildet wird, die Anzahl der verschiedenen möglichen Wörter immens. Ein Beispiel hierfür wäre das Wort „Bundeskanzleramtssprecher“

Während bei neuronalen Netzen das Problem der wenigen Trainingsbeispiele in einem bestimmten Kontext durch die kontinuierliche Repräsentation der Wörter des Kontextes über ihre Merkmalsvektoren abgeschwächt wird, bleibt dieses Problem bei der Ausgabe erhalten, da hier nur die Wahrscheinlichkeiten einzelner Wörter betrachtet werden. So werden auch bei einem großen Trainingstext nur sehr wenige Trainingsbeispiele auf selten vorkommende Wörter abfallen, was bei der Betrachtung des gesamten Vokabulars einer Sprache die Qualität des neuronalen Netzes beschränkt.

Daher bietet es sich an, für die weniger häufig vorkommenden Wörter des Vokabulars einen klassenbasierten Ansatz zu verwenden. Dabei werden diese Wörter in verschiedene Klassen aufgeteilt und das neuronale Netz lernt sowohl die Wahrscheinlichkeit des Auftretens dieser Klassen als auch die Wahrscheinlichkeit des Auftretens der häufigsten Wörter des Vokabulars. Damit werden die gerade erläuterten Probleme umgangen. Die Parameter des neuronalen Netzes steigen nur mit der Anzahl der Klassen an und die Matrixmultiplikationen bleiben dadurch performant. Zudem sind für jede Klasse deutlich mehr Trainingsbeispiele vorhanden als für die einzelnen Wörter der Klasse alleine.

4. Architektur des neuronalen Netzes

Das in dieser Arbeit für das Sprachmodell verwendete neuronale Netz besteht aus einer Eingabeschicht, mehreren verdeckten Schichten und einer Ausgabeschicht. Dabei wurden im Gegensatz zu der in [Schw07] beschriebenen Architektur drei verdeckte Schichten verwendet. Da bei diesem Ansatz, wie in Abbildung 2.3 dargestellt, jedes Neuron einer Schicht mit jedem Neuron der nächsten Schicht verbunden ist, kann auch von einem neuronalen *feedforward* Netz gesprochen werden.

Im Folgenden wird die Architektur des neuronalen Netzes anhand der verwendeten Schichten genauer beschrieben. Dabei werden in der Ausgabeschicht, mit deren Hilfe die A-posteriori-Wahrscheinlichkeiten des Auftretens eines Wortes des Vokabular berechnet werden können, einzelne Wörter des Vokabulars, Wortklassen oder Unterklassen betrachtet. Daher wird im Abschnitt 4.4 neben der dafür nötigen Struktur der Ausgabeschicht auch auf die sich daraus ergebenden Möglichkeiten zur Berechnung der Wahrscheinlichkeit des Auftretens eines einzelnen Wortes eingegangen.

4.1 Merkmalsvektoren

Jedes Neuron der Eingabeschicht des neuronalen Netzes erwartet eine reelle Zahl als Eingabe. Daher wird nicht der Index eines Wortes als Eingabe betrachtet, sondern der Merkmalsvektor des entsprechenden Wortes. Die Merkmalsvektoren werden von dem word2vec Tool, welches in Kapitel 6 genauer beschrieben ist, bereits vor dem Training des neuronalen Netzes erlernt. Für den Zugriff auf die Merkmalsvektoren wird die sogenannte *1 of n* Codierung verwendet. Dafür werden die Merkmalsvektoren zeilenweise in einer Matrix gespeichert. Als Eingabe für das neuronale Netz wird dann anstelle des Wortes die Zeile verwendet, in dem der Merkmalsvektor des Wortes gespeichert ist. Die entsprechenden Merkmalsvektoren werden dann ausgewählt und auf die Eingabeschicht projiziert.

4.2 Eingabeschicht

Nun erhält die Eingabeschicht abhängig von der Länge des Kontextes $|h|$ somit $|h|$ Merkmalsvektoren. Die Größe der Eingabeschicht ist folglich abhängig von der

Länge des betrachteten Kontextes und von der Dimension der Merkmalsvektoren. Somit besteht die Eingabeschicht aus $d \cdot |h|$ Neuronen. Als Aktivierungsfunktion für die Neuronen sollte eine stetig differenzierbare Funktion verwendet werden, deren Ableitung nicht konstant 0 ist. Daher kommt die Vorzeichenfunktion nicht in Frage. Da bei der Aktualisierung der Gewichte des neuronalen Netzes die Ableitung der Aktivierungsfunktion verwendet wird, würden bei der Vorzeichenfunktion die Gewichte sich nicht verändern. Zudem muss als Aktivierungsfunktion eine nicht lineare Funktion benutzt werden, da sonst das neuronale Netz nur eine lineare Funktion repräsentieren könnte. Die in der Eingabeschicht verwendete Funktion ist die Sigmoid Funktion:

$$sig(x) = \frac{1}{1+e^{-x}}$$

Wie in Abbildung 4.1 dargestellt, erzeugt diese Funktion für jedes Neuron Werte zwischen 0 und 1. In Abbildung 4.1 sind die zwei beschriebenen Aktivierungsfunktionen dargestellt.

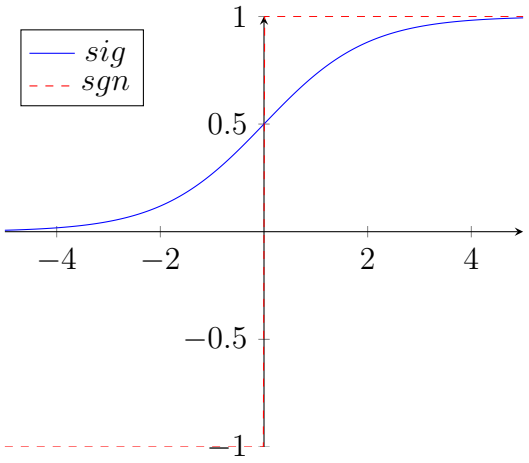


Abbildung 4.1: Mögliche Aktivierungsfunktionen: Sigmoid sig und Vorzeichenfunktion sgn

4.3 Verdeckte Schichten

Auf die Eingabeschicht folgen dann mehrere verdeckte Schichten. Ohne diese Schichten wäre es ebenfalls nicht möglich auch nicht lineare Funktionen zu approximieren. Nach [HoSW89] kann ein neuronales feedforward Netz bereits mit einer verdeckten Schicht quasi alle möglichen Funktionen approximieren, vorausgesetzt die verdeckte Schicht besitzt genügend Neuronen. Die verdeckten Schichten verwenden genauso wie die Eingabeschicht die Sigmoid Funktion als Aktivierungsfunktion.

Die Eingabe der verdeckten Schicht ist abhängig von der Ausgabe der vorherigen Schicht $a_{i-1} \in \mathbb{R}^m$, den Gewichten zwischen den Neuronen der beiden Schichten $W \in \mathbb{R}^{m \times n}$ und dem Bias $b \in \mathbb{R}^n$. Dabei bezeichnet m die Anzahl der Neuronen der vorherigen Schicht und n die Anzahl der Neuronen der gerade betrachteten Schicht:

$$a = sig(a_{i-1}^T \cdot W + b)$$

Hierbei wird die Sigmoid Funktion komponentenweise auf die Elemente des Vektors angewandt.

4.4 Ausgabeschicht

Die Aufgabe der Ausgabeschicht ist es die Wahrscheinlichkeiten des Auftretens bestimmter Wörter zu berechnen. Um dies umzusetzen, repräsentiert jedes Neuron dieser Schicht je ein Wort oder eine Wortklasse. Als Aktivierungsfunktion wird dabei die Softmax Funktion verwendet:

$$softmax(x)_i = \frac{e^{x_i}}{\sum_{n=1}^{N} e^{x_n}}$$

Hierbei steht N für die Anzahl der Neuronen in der Ausgabeschicht. Durch diese Funktion wird jedem Neuron eine Wahrscheinlichkeit zwischen 0 und 1 zugewiesen. Zudem entsteht aufgrund der Normierung in dieser Funktion eine Wahrscheinlichkeitsfunktion, da sich alle Wahrscheinlichkeiten zu 1 summieren.

Äquivalent zu den verdeckten Schichten berechnet sich hier nun die Ausgabe abhängig von der Ausgabe a der letzten verdeckten Schicht, den Gewichten W und den Bias b zwischen den beiden Schichten:

$$P(w_i|h) = softmax(a^T \cdot W + b)$$

In der Grundimplementierung wurde mit dem neuronalen Netz nur eine Teilmenge des Vokabulars betrachtet. Die Liste der Wörter dieser Teilmenge wird im folgenden mit dem Begriff Shortlist bezeichnet. Die Ausgabe des Netzes berechnet somit für alle betrachteten Wörter $w \in Shortlist$ die Wahrscheinlichkeit des Auftretens $P(w|h)$ des Wortes gegeben seines Kontextes h. Die Wahrscheinlichkeiten der Wörter der Shortlist werden dabei mit einem N-Gramm Sprachmodell P_B mit dem Parameter α_s interpoliert. Da das Sprachmodell des neuronalen Netzes P_N nur die Wahrscheinlichkeit einer Teilmenge der Wörter des Vokabulars bestimmt, wird bei der Berechnung der Ausgabe zudem eine Normalisierung benötigt. Folglich wird durch folgende Formel die Ausgabewahrscheinlichkeit eines Wortes der Shortlist berechnet:

$$P(w|h) = \begin{cases} P_N(w|h) \cdot \alpha_s \cdot P_s + (1 - \alpha_s) \cdot P(w|h) & \text{falls } w \in Shortlist \\ P_B(w|h) & \text{sonst} \end{cases}$$

$$P_s = \sum_{w \in Shortlist} P_B(w)$$

Im Folgenden werden nun verschiedene Möglichkeiten vorgestellt, um für alle Wörter eines großen Vokabulars die Wahrscheinlichkeiten mit dem neuronalen Netz zu betrachten, ohne dass die Auswertung des Sprachmodells zu viel Rechenzeit in Anspruch nimmt.

4.4.1 Ausgabeschicht mit Klassen

Ein Ansatz dafür ist es, die noch nicht betrachteten Wörter in Klassen aufzuteilen. Im Folgenden bezeichnet $c(w)$ eine Funktion, die einem Wort w eine Klasse zuordnet. Nun betrachtet das neuronale Netz neben den Wörtern der Shortlist auch die Klassen $c_1, \ldots, c_n$. Dafür wird die Ausgabeschicht um die Anzahl der Klassen vergrößert.

Eine andere Möglichkeit besteht darin, für die Klassen eine eigene Ausgabeschicht zu benutzen.

Die Wahrscheinlichkeit des Auftretens wird dann sowohl für die Klassen $P(c_i|h)$ als auch für die Wörter der Shortlist gegeben des Kontextes h mit dem neuronalen Netz bestimmt. Um für die Wörter, die in Klassen aufgeteilt wurden, diese Wahrscheinlichkeit zu bestimmen, wird zusätzlich die Wahrscheinlichkeit des Wortes $P(w|c(w))$ gegeben seiner Klasse verwendet. Diese Wahrscheinlichkeit kann zum Beispiel folgendermaßen über die Unigramm-Wahrscheinlichkeit mit einem Trainingstext approximiert werden:

$$P_{Unig}(w_i|c(w)) = \frac{\#w_i}{\sum\limits_{w \in c(w_i)} \#w}$$

Hierbei bezeichnet $\#w$ die Anzahl des Auftretens des Wortes w in dem Trainingstext.

Bei der Berechnung der Wahrscheinlichkeiten der Wörter kann dann wieder zwischen dem Sprachmodell des neuronalen Netzes und einem N-Gramm Sprachmodell unterschieden werden. Dabei können zwei Interpolationsparameter verwendet werden. Der Parameter α_s gibt das Gewicht des Sprachmodells des neuronalen Netzes bei der Betrachtung der Wörter der Shortlist an. Für die Gewichtung der Wahrscheinlichkeiten der restlichen Wörter wird der Interpolationsparameter α_r verwendet. Aufgrund der verschiedenen Parameter wird die Summe der Wahrscheinlichkeiten der Wörter der Shortlist und der restlichen Wörter sowohl für das N-Gramm Sprachmodell als auch für das Sprachmodell des neuronalen Netzes benötigt. In der folgenden Formel bezeichnet P_{B_s} und P_{B_r} die Summe der Wahrscheinlichkeit der Wörter der Shortlist und der restlichen Wörter für das N-Gramm Sprachmodell. Analog dazu bezeichnet P_{N_s} und P_{N_r} die entsprechenden summierten Wahrscheinlichkeiten für das Sprachmodell des neuronalen Netzes:

$$\begin{aligned}
P_{B_s} &= \sum_{w \in Shortlist} P_B(w) \\
P_{B_r} &= \sum_{w \notin Shortlist} P_B(w) = 1 - P_{B_s} \\
P_{N_s} &= \sum_{w \in Shortlist} P_N(w) \\
P_{N_r} &= \sum_{w \notin Shortlist} P_N(w) = 1 - P_{N_s}
\end{aligned}$$

Wenn für die Klassen und für die Wörter der Shortlist eine Ausgabeschicht verwendet wird, berechnen sich die Wahrscheinlichkeiten dann folgendermaßen:

$$P(w|h) = \begin{cases} \frac{P_N(w|h)}{P_{N_s}} \cdot \alpha_s \cdot P_{B_s} + P_B(w|h) \cdot (1 - \alpha_s) & \text{falls } w \in Shortlist \\ \frac{P_N(c(w)|h)}{P_{N_r}} \cdot P_{Unig}(w|c(w)) \cdot \alpha_r \cdot P_{B_r} + P_B(w|h) \cdot (1 - \alpha_r) & \text{sonst} \end{cases}$$

Bei diesem Ansatz verändert sich somit die Summe der Wahrscheinlichkeiten der Wörter der Shortlist im Vergleich zum N-Gramm Sprachmodell nicht. Es gilt also:

$$\sum_{w \in Shortlist} P(w|h) = \sum_{w \in Shortlist} P_B(w|h)$$

Wenn die beiden Interpolationsparameter den gleichen Wert haben, ist es möglich diese Einschränkung aufzuheben, indem die Wahrscheinlichkeiten folgendermaßen berechnet werden:

$$P(w|h) = \begin{cases} P_N(w|h) \cdot \alpha + P_B(w|h) \cdot (1-\alpha) & \text{falls } w \in Shortlist \\ P_N(c(w)|h) \cdot P_{Unigr}(w|c(w)) \cdot \alpha + P_B(w|h) \cdot (1-\alpha) & \text{sonst} \end{cases}$$

4.4.2 Ausgabeschichten mit Klassen und Unterklassen

Eine weitere Möglichkeit besteht darin, auch die Wahrscheinlichkeit $P(w|c(w))$ mit dem neuronalen Netz zu approximieren. Dafür wird eine weitere Klassenzuordnung verwendet. Wie im vorherigen Ansatz beschrieben, wird eine Zuordnung $c_o(w)$ von einem Wort zu seiner Oberklasse verwendet. Zusätzlich wird danach eine zweite Zuordnung $c_u(w)$, die ein Wort einer Unterklasse zuordnet, gebildet. Für diese zweite Zuordnung gilt, dass die Wörter jeder Oberklasse in jeweils verschiedene Unterklassen aufgeteilt werden und somit keine zwei Wörter sowohl der gleichen Klasse als auch der gleichen Unterklasse zugeordnet sind. Diese Bedingungen lassen sich wie folgt formulieren:

$$\forall w_1, w_2 \notin \text{Shortlist} \wedge w_1 \neq w_2 : (c_o(w_1) \neq c_o(w_2)) \vee (c_u(w_1) \neq c_u(w_2))$$

Für die Unterklassen wird nun eine separate Ausgabeschicht benötigt. Diese verwendet wieder die Softmax Funktion als Aktivierungsfunktion und ist mit einer Matrix aus Gewichten mit der letzten verdeckten Schicht verbunden. Die Architektur des dadurch entstehenden neuronalen Netzes ist in Abbildung 4.2 dargestellt. Die zweite Ausgabeschicht berechnet nun also $P(c_u(w)|h)$. Aufgrund der Tatsache, dass keine zwei Wörter derselben Ober- und Unterklasse zugeordnet sind, gilt somit für die Wörter jeder Oberklasse o:

$$\sum_{w \in o} P(c_u(w)|h) \leq 1$$

Aufgrund der geforderten Bedingungen an die Unterklassenzuordnungen ist es nun jedoch möglich, die Wahrscheinlichkeiten des neuronalen Netzes anstatt mit der Unigramm-Wahrscheinlichkeit innerhalb der Oberklasse mithilfe der Unterklasse zu berechnen:

$$P_N(w_i|h) = P(c_o(w_i)|h) \cdot \frac{P(c_u(w_i)|h)}{\sum\limits_{w \in c_u(w_i)} P(c_u(w)|h)} \qquad \text{falls } w \notin Shortlist$$

Da sich diese Wahrscheinlichkeit nicht für jede mögliche Oberklasse zu eins addiert, wurde die Wahrscheinlichkeit der Unterklasse in der Formel normiert. Bei Verwendung der Interpolationsparameter α_s und α_r berechnet sich damit die Wahrscheinlichkeit analog zum vorherigen Ansatz:

$$P(w|h)= \begin{cases} \frac{P_N(w|h)}{P_{N_s}} \cdot \alpha_s \cdot P_{B_s} + P_B(w|h) \cdot (1-\alpha_s) & \text{falls } w \in Shortlist \\ \frac{P_N(c(w)|h)}{P_{N_r}} \cdot \alpha_r \cdot P_{B_r} + P_B(w|h) \cdot (1-\alpha_r) & \text{sonst} \end{cases}$$

Außerdem besteht die Möglichkeit für die Klassen und für die Shortlist eine eigene Ausgabeschicht zu erstellen. Durch die verschiedenen Ausgabeschichten werden unterschiedliche Trainingsmethoden für die Wörter der Shortlist und die Klassen ermöglicht, welche in Kapitel 5 genauer erläutert werden. Aufgrund der nun getrennten Ausgabeschichten summieren sich die Wahrscheinlichkeiten der Klassen und die Wahrscheinlichkeiten der Wörter der Shortlist nun jeweils zu 1. Damit entfällt die Normalisierung dieser Wahrscheinlichkeiten und die Formel ändert sich wie folgt:

$$P(w_i|h)= \begin{cases} P_N(w_i|h) \cdot \alpha_s \cdot P_s + P_B(w_i|h) \cdot (1-\alpha_s) & \text{falls } w \in Shortlist \\ P_N(c(w)|h) \cdot \alpha_r \cdot P_r + P_B(w|h) \cdot (1-\alpha_r) & \text{sonst} \end{cases}$$

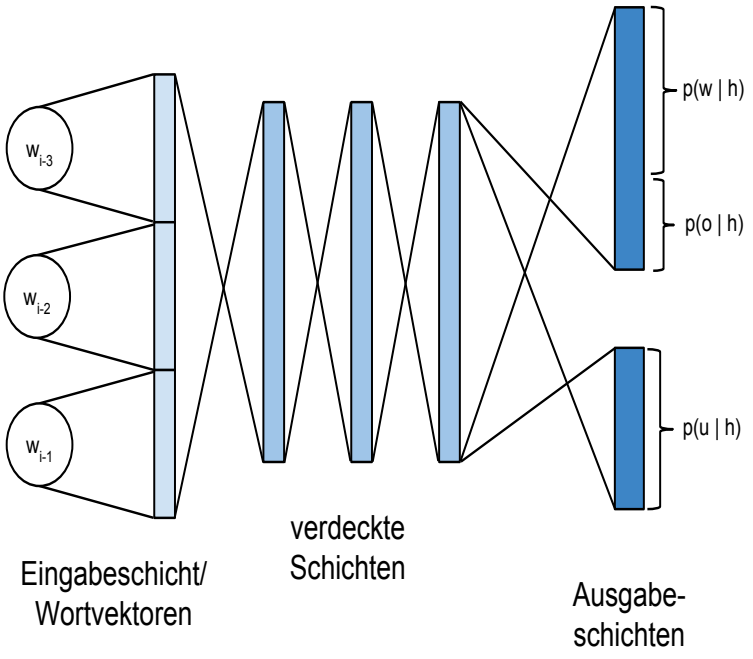


Abbildung 4.2: Architektur des neuronalen Netzes mit zwei Ausgabeschichten

4.5 Zusammenfassung

Das in dieser Arbeit zur Sprachmodellierung benutzte Netz ist ein neuronales *feed-forward* Netz. Einige wichtige Parameter der verwendeten Architektur sind in Tabelle 4.1 dargestellt. Das neuronale Netz besteht aus einer Eingabeschicht mit 600

Anzahl der verdeckten Schichten	3
Neuronen pro verdeckte Schicht	600
Neuronen der Eingabeschicht	600
Länge des Kontextes	3
Dimensionen der Merkmalsvektoren	200

Tabelle 4.1: Parameter des neuronalen Netzes

Neuronen. Auf diese Eingabeschicht werden Merkmalsvektoren von drei Wörtern projiziert. Diese Vektoren werden in einem separaten Schritt erlernt und haben 200 Dimensionen. Auf die Eingabeschicht folgen nun drei verdeckte Schichten mit jeweils 600 Neuronen.

Der letzte Teil des neuronalen Netzes ist nun für die Berechnung der Wahrscheinlichkeiten zuständig und besteht aus mindestens einer Ausgabeschicht, welche für jede betrachtete Wahrscheinlichkeit ein Neuron benötigt. Bei einem ersten Modell werden mit dem neuronalen Netz nur eine Teilmenge des Vokabulars, welche als Shortlist bezeichnet wird, betrachtet. Alle Wörter dieser Teilmenge bekommen in einer Ausgabeschicht jeweils ein Neuron zugewiesen.

Ein weiteres Modell betrachtet das gesamte Vokabular und teilt dabei die Wörter in Klassen auf. Die Wahrscheinlichkeit des Auftretens der Klasse eines Wortes wird dann durch das neuronale Netz betrachtet. Mithilfe der Unigrammwahrscheinlichkeit des Wortes in dieser Klasse kann dann die Wahrscheinlichkeit des Auftretens des Wortes berechnet werden. Die Wörter der Shortlist werden jedoch weiterhin einzeln betrachtet. Bei diesem Modell können die Wahrscheinlichkeiten der Wörter und der Klassen zusammen in einer Schicht betrachtet werden, indem der Schicht für jede Klasse ein weiteres Neuron zugewiesen wird. Alternativ kann auch eine weitere Ausgabeschicht, die nur für die Klassen zuständig ist, verwendet werden.

Außerdem kann über eine weitere Ausgabeschicht auch die Wahrscheinlichkeit des Auftretens eines Wortes gegeben seiner Klasse berechnet werden. Dafür werden die Worte zusätzlich in Unterklassen gegliedert, so dass jedes Wort einer Klasse einer anderen Unterklasse zugeordnet ist. Die Unterklassen ersetzen damit die Unigramm-Wahrscheinlichkeit innerhalb einer Klasse. Die dafür benötigte zusätzliche Ausgabeschicht besitzt dabei wiederum pro Unterklasse ein Neuron.

5. Training des neuronalen Netzes

Nachdem in Kapitel 4 die Architektur und die Auswertung eines neuronalen Netzes beschrieben wurde, beschäftigt sich dieses Kapitel damit, die optimalen Gewichte der Verbindungen zwischen den Neuronen des Netzes zu finden. Dieser Prozess wird auch als die Trainingsphase bezeichnet.

In dem Rest dieses Kapitels werden die folgenden Notationen verwendet:

- l bezeichnet die Anzahl der Schichten des neuronalen Netzes
- n_j bezeichnet die Anzahl der Neuronen einer Schicht mit $1 \geq j \geq l$
- T bezeichnet eine Menge an Trainingsbeispielen
- $(x, y) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_l}$ ist ein Trainingsbeispiel aus T
- W bezeichnet die Gesamtheit der Gewichte eines neuronalen Netzes
- $o_i(x)$ bezeichnet die Ausgabe des Neurons i gegeben dem Eingabevektor x
- $x^{(i)}$ bezeichnet das i. Element des Vektors x

5.1 Fehlerfunktionen

Um die optimalen Gewichte zu bestimmen, muss zuerst ein Fehlermaß definiert werden, mit dem die Ausgaben neuronaler Netze mit verschiedenen Gewichten W verglichen werden können. Im Folgenden werden dafür die mittlere quadratische Abweichung (englisch: mean squared error) MSE und die Kreuzentropie (englisch: cross entropy) CE verwendet. Dabei ist die mittlere quadratische Abweichung für ein Trainingsdatum (x, y) folgendermaßen definiert:

$$MSE_{(x,y)}(W) = \frac{1}{n_l} \cdot \sum_{i=1}^{n_l} (y^{(i)} - o_i(x))^2$$

Die Kreuzentropie ist dagegen definiert über:

$$CE_{(x,y)}(W) = \sum_{i=1}^{n_l} -y^{(i)} \log(o_i(x)) - (1 - y^{(i)}) \log(1 - o_i(x))$$

	erste Schicht	weitere Schichten
corruption	0.3	0.2
Fehlerfunktion	MSE	CE
Lernrate	0,001	0,01
Größe der Bündel	128	128
Anzahl der Bündel	600 000	400 000

Tabelle 5.1: Parameter der verdeckten Schichten

5.2 Initialisierung

Bevor das eigentliche Training des neuronalen Netzes beginnt, werden zuerst geeignete Initialwerte für die Gewichte des neuronalen Netzes gesucht. Dafür wurde das in [VLBM08] vorgeschlagene Verfahren eines *stacked denoising autoencoders* verwendet. Dabei werden nacheinander die Gewichte zwischen zwei Schichten so gewählt, dass das neuronale Netz möglichst gut Trainingsbeispiele, die zusätzlich verrauscht worden sind, rekonstruieren kann. Die Eingabe der $(k + 1)$-ten Schicht entspricht dabei jeweils der Ausgabe der k-ten Schicht. Eine bestimmte, von dem *corruption*-Parameter abhängige, Anzahl an Eingaben der ersten Schicht wird dabei verändert. Dabei bezeichnet *corruption* üblicherweise die Wahrscheinlichkeit, dass ein Merkmal der Eingabe verändert wird.

Da das Aktualisieren der Gewichte Rechenzeit benötigt und dies daher nicht nach jedem einzelnen Trainingsdatum erstrebenswert ist, werden die Trainingsdaten dabei in gleich große Bündel aufgeteilt. Alle Trainingsbeispiele dieses Bündels werden ausgewertet und danach wird anhand der Fehlerfunktion ein Fehler berechnet. Im Anschluss erfolgt das Aktualisieren der Gewichte des neuronalen Netzes. Dabei gibt eine davor festgelegte Lernrate an, wie stark die Gewichte verändert werden, um eine Verbesserung des Fehlers für die gesehenen Beispiele zu erreichen. Ein Vorteil einer kleinen Bündelgröße ist es, dass die Gewichte häufiger aktualisiert werden, eine größere Bündelgröße führt dahingegen zu bedeutungsvolleren Gewichtsaktualisierungen.

Für die Initialisierung der verdeckten Schichten wurden die in Tabelle 5.1 aufgelisteten Parameter verwendet.

5.3 Training

Nach dem Abschluss dieser Phase wird dem neuronalen Netz noch eine Ausgabeschicht hinzugefügt und die eigentliche Trainingsphase beginnt. Dabei werden nun die Gewichte mithilfe des *Backpropagation*-Algorithmus optimiert. Der Gedanke dieses Algorithmus ist es, den Fehler von Schicht zu Schicht auf die Gewichte zurückzuführen, die einen Einfluss auf die fehlerhafte Ausgabe hatten. Wie in [Mitc97] beschrieben, werden dabei diese Gewichte dann abhängig von einem Lernparameter verändert, so dass bei denselben Trainingsbeispielen der Fehler zukünftig kleiner ausfällt. Dieses Verfahren ist ein Spezialfall des Gradientenverfahrens.

Das Training ist in verschiedene Epochen aufgeteilt, wobei das neuronale Netz in jeder Epoche mit den gesamten Trainingsdaten trainiert wird. Nach jeder Epoche

wird der Fehler auf den sogenannten Validierungsdaten getestet, um zu überprüfen, ob die Änderungen der Gewichte immer noch einen positiven Einfluss auf die Validierungsdaten haben. Dadurch wird eine Überanpassung der Gewichte an die Trainingsdaten vermieden.

In jeder Epoche kann dabei die Lernrate verändert werden. Für die Festlegung der Lernrate wurde das in [GKKC07] beschriebene *newbob*-Verfahren verwendet. Dieses Verfahren startet mit einer Lernrate von α und behält diese bei, bis der auf den Validierungsdaten gemessene Fehler sich um weniger als θ_1 verringert. Danach wird die Lernrate üblicherweise halbiert. Das Training wird beendet, wenn der Fehler sich um weniger als θ_2 verringert, nachdem die Lernrate im vorherigen Schritt herabgesetzt wurde. In dieser Arbeit wurde eine initiale Lernrate von 1 gewählt und $\theta_1 = 0,005$ und $\theta_2 = 0,0001$ gesetzt.

5.4 Trainingsdaten

Die Trainingsdaten wurden aus verschiedenen Quellen ausgewählt. Unter anderem wurden Zeitungstexte, Texte und Kommentare von verschiedenen Internetseiten von Nachrichtenmagazinen, transkribierte Reden deutscher Präsidenten und Kanzler und übersetzte TED[1] Vorträge verwendet. Insgesamt enthielten die Texte 1,7 Milliarden Wörter. Die verwendeten Trainingsdaten entsprechen den in [KHMS+] zur Erstellung eines Sprachmodells verwendeten deutschen Texten.

Da es für das Training des neuronalen Netzes zu zeitaufwendig ist alle Texte als Trainingsdaten zu benutzen, wurde nur ein Teil daraus ausgewählt. Dafür wurden die verschiedenen Texte mithilfe eines Verfahrens des maschinellen Lernens nach ihrer Relevanz gewichtet. Danach wurden, entsprechend ihrer Gewichtung, zufällig Sätze aus den jeweiligen Korpus ausgewählt und dem Trainingstext hinzugefügt. Da dabei die Sätze von sehr hoch gewichteten Texten potentiell öfter ausgewählt werden konnten, wurde die maximale Anzahl, mit der ein Satz im Trainingstext vorkommen darf, beschränkt.

Für die Grundimplementierung des neuronalen Netzes wurde nach diesem Verfahren ein Trainingstext mit 2 Millionen Sätzen und 34 Millionen Wörtern ausgewählt. Für das neuronale Netz sind aus diesem Trainingstext nur Beispiele relevant, die auch in der Shortlist vorkommen. 91,3% der Wörter des Trainingstextes waren dabei in der verwendeten Shortlist, die aus knapp über 18 000 Wörter besteht, enthalten. Damit wurden die tatsächlich verwendeten Beispiele auf knapp über 32 Millionen Wörter beschränkt.

Bei der Verwendung einer Ausgabeschicht, die neben den Wörtern der Shortlist auch die Klassen der restlichen Wörter betrachtet, wurde derselbe Trainingstext verwendet. Da nun nicht nur die Wörter der Shortlist relevant für das neuronale Netz sind, sondern auch alle anderen Wörter des Vokabulars, wurden jetzt 98,3% der Wörter als Trainingsbeispiele verwendet. Damit fiel auf die Neuronen, die die Klassen repräsentieren, 7% der Beispiele ab. Im Gegensatz dazu wurden die Neuronen der Shortlist mit 91,3% des Trainingstextes trainiert.

[1] TED bezeichnet eine ursprünglich jährlich veranstaltete Innovations-Konferenz, deren Vorträge auf http://www.ted.com/ veröffentlicht werden

Text	Wörter	Wörter der Shortlist	restliche Wörter des Vokabulars
t10	34 M	31M	2,4 M
t50	102 M	89 M	10 M
all	271 M	235 M	33 M

Tabelle 5.2: Anzahl der Wörter der verschiedenen Trainingstexte in Millionen (M)

Aufgrund dieser Tatsache wurden für die neuronalen Netze mit mehreren Ausgabeschichten auch Verfahren evaluiert, die mehr Trainingsbeispiele für die Klassen und Unterklassen bereitstellen.

Für das hierarchische Modell, das eine Ausgabeschicht für die Shortlist und die Klassen und eine weitere Schicht für die Unterklassen enthält, kann die Schicht der Unterklassen ohne Probleme mit einem größeren Trainingstext trainiert werden. Falls bei dem Trainingstext der Ausgabeschicht für die Shortlist und die Klassen künstlich mehr Trainingsbeispiele für die in Klassen aufgeteilten Wörter hinzugefügt werden, wird die Wahrscheinlichkeitsverteilung jedoch in Richtung dieser Wörter verschoben.

Um auch die Klassen sinnvoll mit einer größeren Trainingsmenge zu trainieren, wurde, wie in Kapitel 4 beschrieben, die Ausgabeschicht aufgeteilt. Damit kann nun auch die Schicht für die Klassen mit einem größeren Trainingstext trainiert werden. Die Wörter der Shortlist wurden jeweils mit demselben Trainingstext trainiert. Für die Klassen und Unterklassen ist dagegen auch der dreimal so große Text t50 verwendet worden. Zusätzlich entspricht bei dem Trainingstext *all* die Anzahl der Beispiele der restlichen Wörter des Vokabulars der Anzahl der Wörter der Shortlist des Trainingstextes t10. Die Anzahl der Trainingsbeispiele der verwendeten Texte ist in Tabelle 5.2 dargestellt.

6. Implementierung

In diesem Kapitel wird zuerst das für die Implementierung der Spracherkennung verwendete Janus Recognition Toolkit[1] vorgestellt. Danach wird die Software vorgestellt, die für die Umsetzung des neuronalen Netzes verwendet wurde. Dabei wird neben der Erstellung der Merkmalsvektoren der einzelnen Wörter des Vokabulars auch auf die eigentliche Implementierung des neuronalen Netzes eingegangen. In einem abschließenden Abschnitt werden dann die Clusteringverfahren, die für die Erstellung der Worklassen verwendet wurden, genauer erläutert.

6.1 Spracherkennung

Das Janus Recognition Toolkit (kurz: Janus) wurde in dieser Arbeit für die Implementierung der Spracherkennung benutzt. Janus wurde im Interactive System Lab des Karlsruher Instituts der Technologie und der Carnegie Mellon Universität entwickelt und unter anderem, wie in [FuWK07] beschrieben, zur simultanen Übersetzung von Vorlesungen verwendet. Janus ist in der Programmiersprache C programmiert und implementiert eine *Tcl/Tk*-Schnittstelle. *Tcl* ist dabei eine Open-Source-Skriptsprache und *Tk* ein Toolkit zur Programmierung von grafischen Benutzeroberflächen.

Janus unterstützt viele Audioformate und implementiert moderne Techniken zur akustischen Vorverarbeitung. Für die akustische Modellierung werden Algorithmen wie das Viterbi-Training und das EM-Training unterstützt. Um die vorgestellten Methoden mit neuronalen Netzen zu implementieren, wurde die Schnittstelle zur Sprachmodellierung von Janus angepasst.

6.2 Neuronales Netz

Für die Eingabe des neuronalen Netzes mussten zuerst für alle Wörter des Vokabulars Merkmalsvektoren erstellt werden. Diese wurden in einem seperaten Schritt mit dem word2vec[2] Tool erlernt, das in [MCCD13a] vorgestellt wurde. Dabei werden

[1] http://isl.anthropomatik.kit.edu/english/1406.php
[2] https://code.google.com/p/word2vec/

```
import theano
# deklarieren einer symbolischen Variable
a = theano.tensor.vector("a")
# Aufbau eines symbolischen Ausdrucks
b = a + a ** 10
# Kompilieren der Funktion
f = theano.function([a], b)
# gibt "array([0,2,1026])" aus
print f([0, 1, 2])
```

Abbildung 6.1: Deklarieren und Auswerten einer symbolischen Funktion mit Theano[4]

aus einem Trainingstext für alle vorkommenden Wörter N-dimensionale Merkmalsvektoren erstellt. Dazu wird ein einfacheres Modell als das in Kapitel 2 vorgestellte Modell des neuronalen Netzes verwendet, bei dem keine verdeckte Schicht vorhanden ist. Durch diese Vereinfachung ist es möglich die Merkmalsvektoren effizient auf deutlich größeren Daten zu trainieren. Außerdem werden hier nicht nur die vor einem Wort vorkommenden Wörter als Eingabe betrachtet, sondern auch die Wörter, die dem Wort, das gerade klassifiziert werden soll, nachfolgen.

Für die eigentliche Implementierung des neuronalen Netzes wurde dann Theano[3] verwendet. Theano ist eine Python Bibliothek, die seit 2007 an der Universität Montreal entwickelt wird. Wie in [BBBL+10] beschrieben, verwendet Theano die Syntax von Numpy und ist im Gegensatz zu Python statisch typisiert. In Theano können mathematische Ausdrücke symbolisch angegeben und dann zu einem späteren Zeitpunkt ausgewertet werden. Ein Beispiel hierfür wird in Abbildung 6.1 dargestellt. Aus den symbolischen Repräsentationen der Ausdrücke wird dann ein Berechnungsgraph aufgebaut. Da die Ausdrücke rein funktional sein müssen, also keine Seiteneffekte haben dürfen, kann dieser Graph dann stark optimiert werden. Zum Beispiel können bei der Optimierung numerisch instabile Ausdrücke ersetzt werden und identische Funktionen mit gleicher Eingabe erkannt werden. Der optimierte Berechnungsgraph wird dann benutzt um C Code für die CPU oder CUDA Code für die GPU zu generieren. Zudem unterstützt Theano viele mathematische Funktionen, ist in der Lage Gradienten zu berechnen und enthält bereits Methoden, die die Sigmoid und die Softmax Funktion implementieren. Aufgrund dieser Eigenschaften ist Theano für die Implementierung von neuronalen Netzen attraktiv. Vor allem die automatische Erzeugung von CUDA Code ermöglicht deutliche Zeitersparnisse, da hierdurch das neuronale Netz auf der GPU trainiert werden kann.

6.3 Clusteringverfahren

Um die Wörter in Klassen aufzuteilen werden im Folgenden Clusteringverfahren vorgestellt. In dieser Arbeit wird der Fokus auf den K-Means Algorithmus gelegt, der auch in [LOAG+13] zur Aufteilung der Wörter in Klassen und Unterklassen verwendet wurde.

[3] https://github.com/Theano/Theano/

[4] vgl. deeplearning.net/sofware/theano/tutorial/symbolic_graphs.html#optimizations

6.3.1 K-Means

Mit dem K-Means Algorithmus kann eine Menge von Merkmalsvektoren in K verschiedene Klassen aufgeteilt werden. Dabei wird während der Ausführung des Algorithmus die durchschnittliche Distanz zwischen den Punkten einer Klasse und den Klassenzentren minimiert beziehungsweise ihre durchschnittliche Ähnlichkeit maximiert. Bei dem *spärischen* K-Means Algorithmus, der in [DhMo01] bereits erfolgreich für das Clustern von Dokumenten benutzt wurde, wird dabei die sogenannte Kosinus-Ähnlichkeit verwendet. Der Kosinus zwischen zwei Merkmalsvektoren a und b der Dimension n ist dabei folgendermaßen definiert:

$$cos(a,b) = \frac{a \cdot b}{\|a\| \cdot \|b\|} = \frac{\sum_{i=1}^{n} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \cdot \sqrt{\sum_{i=1}^{n} b_i^2}}$$

Dabei ist die Kosinus-Ähnlichkeit zweier Vektoren maximal, wenn die Vektoren in die selbe Richtung zeigen und minimal, wenn sie in die genau entgegengesetzte Richtung zeigen. Die Kosinus-Ähnlichkeit nimmt Werte zwischen -1 und 1 an. Die Zentren der Klassen werden wie folgt über die Summe der normierten Merkmalsvektoren einer Klasse K berechnet:

$$\sum_{v \in K} \|v\|$$

Dadurch beeinflusst die Länge der Vektoren nicht ihre Gewichtung zur Berechnung des Zentrums ihrer Klasse.

6.3.2 Clustering mit Größenbeschränkungen

Durch herkömmliche Clusteralgorithmen entstehen häufig Klassen stark unterschiedlicher Größe. In diesem Abschnitt wird nun ein Verfahren vorgestellt, um die Größe der Wortklassen zu beschränken. Dafür wir das in [BaGh] vorgeschlagene Vorgehen erweitert um sowohl eine Mindestgröße als auch auch eine Maximalgröße für die generierten Klassen zu garantieren. Wie beim bereits in Absatz 3.1.5 vorgestellten Ansatz werden dafür drei Schritte benötigt:

Zuerst wird eine Teilmenge der betrachteten Wörter in die K verschiedene Klassen aufgeteilt. Hierfür wird der *spärische* K-Means Algorithmus verwendet. Dabei sollte die für diesen Schritt verwendete Teilmenge der Wörter möglichst gut die gesamte Menge der betrachteten Wörter repräsentieren. Deshalb werden die am häufigsten im Trainingstext vorkommenden Wörter, die nicht schon in der Shortlist enthalten sind, als Teilmenge verwendet.

In einem zweiten Schritt wird dann jede Klasse, die noch nicht eine Mindestanzahl an Wörtern enthält, bis zu der gegebenen Mindestgröße aufgefüllt. Dafür gilt wiederum, dass jeder Punkt der ihm ähnlichsten Klasse zugewiesen wird oder alle Klassen, die ihm ähnlicher sind als die ihm zugewiesene Klasse, bereits ihre Mindestgröße erreicht haben. Die restlichen noch nicht zu einer Klasse zugewiesenen Wörter werden nun so auf die Klassen aufgeteilt, dass jedes Wort der ihm ähnlichsten Klasse zugewiesen wird, sofern diese noch nicht eine Maximalgröße erreicht hat.

Nun sind alle Wörter einer Klasse zugeordnet und die Klassenzuordnungen werden durch Wechsel der Wörter in andere Klassen noch verbessert. Dafür werden in jeder Iteration die Wörter, sofern dies keine der verlangten Größenbeschränkungen verletzt, solange einer Klasse mit einem ähnlicheren Klassenzentrum zugewiesen bis sich die Klassenzuordnungen nicht mehr signifikant ändern.

6.3.3 Sekundäres Clustering

Nachdem die Wörtern in Klassen aufgeteilt worden sind, werden sie nun in Unterklassen unterteilt. Dabei wird eine Zuordnung gesucht, in dem jedes Wort einer Klasse einer jeweils anderen Unterklasse zugeordnet wird. Zudem werden n verschiedene Unterklassen gebildet, wobei n die Anzahl der Wörter der größten Klasse ist.

Der Algorithmus 6.2 erhält als Eingabe eine bereits davor erzeugte Klassenzuordnung. Die gegebenen Klassen werden dann zuerst absteigend nach der Anzahl ihrer Wörter sortiert. Die Wörter der größten Klasse werden anfangs jeweils einer Unterklasse zugeordnet, so dass nun jede Unterklasse genau ein Wort enthält. Somit bilden diese Wörter auch gleichzeitig das initiale Klassenzentrum ihrer Klasse.

Nun werden die Wörter der restlichen Klassen auf die Unterklassen aufgeteilt. Dies erfolgt nacheinander für jede Klasse. Für eine Klasse werden jeweils für jede Unterklasse die Wörter der derzeitigen Klasse absteigend nach der Ähnlichkeit zu dem Zentrum der Unterklasse sortiert. Solange noch nicht alle Wörter der Klasse aufgeteilt sind, bittet jede freie Unterklasse um das Wort, das ihr am Ähnlichsten ist und von dem sie noch nicht abgewiesen worden ist. Wenn dieses Wort noch keiner Unterklasse zugeordnet wurde oder die dem Wort zugeordnete Klasse nicht ähnlicher als die um das Wort bittende Unterklasse ist, wird es der neuen Unterklasse zugeordnet. Ansonsten muss die Unterklasse um das ihr nächst ähnlichste Wort bitten. Dies wird für die Wörter jeder Klasse wiederholt, so dass abschließend alle Wörter einer Unterklasse zugeordnet sind.

Danach werden die Wörter noch solange einer ähnlicheren Unterklasse zugeordnet, sofern die Unterklasse noch kein Wort der Klasse des aktuellen Wortes enthält, bis sich die Zuordnung zu den Unterklassen nicht mehr nennenswert verändert.

```
Eingabe: die Liste classes, die für jede Klasse die Menge ihrer Wörter enthält und
         die Anzahl der Wörter N
Ausgabe: eine Zuordnung wordToSubcl, die jedem Wort eine Unterklasse zuordnet

#Sortieren der Klassen nach der Anzahl ihrer Wörter
sortedClasses = sorted(classes, size, desc)

#Initialisieren der Zuordnung wordToSubcl mit -1
wordToSubcl = [-1] * N

#Zuordnungen jedes Wortes der ersten Klasse zu jeweils einer Unterklasse
i = 0
for (word : sortedClass[0]):
   wordToSubcluster[word] = i
   i++

#Zuordnung der Wörter der restlichen Klassen zu einer Unterklasse
for cl in sortedClasses[1:] :
   for subcl in subclasses:
      #Sortieren der Wörter einer Klasse absteigend nach der
      #Ähnlichkeit zu der Unterklasse subcl
      wordsBySimilarityTo[subcl] = sorted(cl, similarityTo(subcl), desc)

   for each (subcl : subclasses):
      subclassFree[subcl] = true

   while (not all words of cl assigned):
      for subcl in subclasses:

         #Falls die Unterklasse schon belegt ist, fahre mit der
         #  nächsten Unterklasse fort
         if (not subclassFree[subcl]):
            continue

         #Unterklasse versucht, dass ihr nächst ähnlichste Wort zu erhalten
         word = pointsBySimilarityTo[subcl].next()

         #das Wort wurde noch keiner Unterklasse zugeordnet
         if (wordToSubcl[word] < 0):
            isAssigned[word] = true
            wordToSubclass[word] = subcl
            subclassFree[subcl] = false

         else:
            #die aktuelle Unterklasse ist dem Wort ähnlicher als die zugeordnete Unterklasse
            if (similarity(word, subcl)) > similarity(word, wordToSubcl[word]):
               subclusterFree[wordToSubcl[word]] = true
               subclusterFree[subcl] = false

               wordToSubcl[word] = subcl
```

Abbildung 6.2: Algorithmus zur Erzeugung von Unterklassen

7. Evaluation

In diesem Kapitel wird zuerst die Qualität der durch die verschiedenen Algorithmen erstellten Wortklassen untersucht und verglichen. Danach werden die in Kapitel 4 vorgestellten Modelle der neuronalen Netze mit den verschiedenen Wortklassen in der Spracherkennung an einem konkreten Beispiel evaluiert.

7.1 Evaluation der Wortklassen

In diesem Abschnitt werden die Ergebnisse der Clustering Algorithmen vorgestellt. Die Aufgabe der Algorithmen besteht darin, die Wörter, die nicht in der Shortlist enthalten sind, in Wortklassen zu unterteilen. Die Anzahl dieser Wörter beläuft sich auf knapp über 232 000. Dabei werden verschiedene Algorithmen betrachtet, die allesamt anhand der Kosinus-Ähnlichkeit der Merkmalsvektoren der Wörter Wortklassen bilden. Die Kosinus-Ähnlichkeit *cosSim* (englisch: cosine similarity) bezeichnet dabei das in Abschnitt 6.3.1 vorgestellte Ähnlichkeitsmaß. Als zu Grunde liegender Clustering Algorithmus wird dabei jeweils der K-Means Algorithmus verwendet. Des Weiteren wird das in Abschnitt 6.3.2 vorgestellte Rahmenwerk zur Bildung von Wortklassen mit Größenbeschränkungen ebenfalls mit dem K-Means Algorithmus verwendet. Nach dem Erstellen der Wortklassen werden, wie in Abschnitt 6.3.3 beschrieben, auf Basis der Wortklassen Unterklassen erstellt.

Für die Algorithmen werden einige Beispiele von Klassenzuordnungen aufgezeigt, um einen Eindruck zu vermitteln, welche Wörter der K-Means Algorithmus der selben Klasse zuordnet. Außerdem werden anhand der durchschnittlichen Kosinus-Ähnlichkeit der Wörter zu ihren Klassen die verschiedenen Clusteringverfahren bewertet. Zudem wird die Größe der durch den K-Means Algorithmus entstehenden Klassen als weiteres Kriterium thematisiert und damit ein Vergleich mit den vorgeschlagenen Clustering Verfahren mit Größenbeschränkungen gezogen.

7.1.1 Klassen

Um einen ersten Eindruck von der Aufteilung der Wörter in verschiedene Klassen zu bekommen, werden in Tabelle 7.1 einige Beispiele dargestellt. Dafür wurden mit dem

Anglisten Betriebswirtschaftler BWLer Informatiker Jurastudenten Juristen Koryphäen Kriminologen Hochschullehrer Koryphäen Linguisten Naturwissenschaftler Philologen plagiieren Promovierte Wirtschaftler Wissenschaftlerinnen Fachkollegen
entgleist entgleiste Gleisen Güterzug Intercity Loks Oberleitung Prellbock Rangierbahnhof Rangierlok Regionalzug Schnellzugs Signalanlagen Unglückszug Waggon Waggons Wagons
Bundesfinanzministerium Finanzdepartement Finanzministerium Finanzverwaltung Staat+ Steuerfahndungs+ Bundesfinanzministerium ZKA BMF
Gigabyte Bit MB USB Festplatten Speicherplatz Arbeitsspeicher Core Memory Megabyte GHz RAM Gigahertz Speicherkapazität Grafikkarte Cache Lieferumfang Terabyte SSD Dual verdoppeltem getaktet erweiterbar Rechenkraft Prozessorleistung Firewire
Sportart Sportarten Judo Karate Rugby Rudern Fechten Volkssport Taekwondo Wrestling Paintball Curling Aikido Snooker Sportliches Kraftsport Männersport Skateboarden Sumo Kricket

Tabelle 7.1: Wörter ausgewählter Klassen, die durch ein 5000-Means Clustering gebildet wurden

K-Means Algorithmus 5 000 Klassen erstellt. Dabei wurden Bezeichnungen für Personengruppen in der Universität (Juristen, Informatiker...) in eine Klasse unterteilt. Unter anderem konnten auch Klassen der Themengebiete Eisenbahn (Güterzug, Signalanlagen, Waggon), Sport (Aikido, Kricket, Volkssport) und mit Computern verbundene Begriffe (RAM, USB, Prozessorleistung) in jeweils eine Klasse unterteilt werden. Auch die Abkürzungen BMF (Bundesfinanzministerium) und ZKA (Zollkriminalamt) konnten erkannt werden und wurden einer Klasse mit dem Thema Steuerpolitik (Bundesfinanzministerium, Bundesfinanzministerium) zugeordnet.

Natürlich finden sich in einer Klasse auch Wörter, für die es weniger Sinn macht, sie für das Erstellen eines Sprachmodells in die gleiche Klasse zu gliedern. Beispiele hierfür sind die thematisch zwar in Bezug stehenden Wörter „Juristen“ und „plagiieren“ beziehungsweise „Prellbock“ und „entgleist“, die aber grammatikalisch nicht die selbe Funktion erfüllen. Für das Wort „verdoppeltem“ gibt es aufgrund der Wortbedeutung auch keine Begründung, warum es in einer Klasse mit IT-Fachbegriffen vorkommt. Es ist aber verständlich, dass dieses Wort aufgrund des Mooreschen Gesetzes oft im Zusammenhang mit Wörtern wie Speicherplatz oder Rechenleistung verwendet wird.

Im Grunde ist anhand der gezeigten Beispiele jedoch zu erkennen, dass mithilfe der Kosinus-Ähnlichkeit auf den Merkmalsvektoren sinnvolle Wortklassen erstellt werden können.

Nachdem durch einige Beispiele ein Einblick in die entstandenen Wortklassen gegeben wurde, werden nun die konkreten Algorithmen mit unterschiedlichen Größenbeschränkungen verglichen. Zuerst wurde ein Clustering ohne Größenbeschränkung mit dem K-Means Algorithmus durchgeführt. Die einzige Einschränkung, die in der Implementierung dieses Algorithmus getroffen wurde, ist, dass jede Wortklasse mindestens ein Wort enthalten muss. Außerdem wurden weiche Größenbeschränkungen

K	Ohne		Weich		Streng	
	Min	Max	Min	Max	Min	Max
500	1	-	100	1 600	400	528
1 000	1	-	50	800	200	264
2 000	1	-	25	400	100	132
5 000	1	-	10	160	40	52
10 000	1	-	5	80	20	26

Tabelle 7.2: Minimal und maximal erlaubte Klassengrößen bei den verwendeten Größenbeschränkungen

definiert. Dabei wurde für die Aufteilung der Wörter in K Klassen eine Mindestgröße von $\frac{50000}{K}$ und eine Maximalgröße von $\frac{800000}{K}$ festgelegt. Bei den strengen Größenbeschränkungen wurden dagegen Klassen gebildet, die zwischen $\frac{200000}{K}$ und $\frac{264000}{K}$ Wörter enthalten.

Die durch diese Einschränkungen entstehenden Größen für die verwendete Klassenanzahl sind Tabelle 7.2 zu entnehmen. Das Clustering mit den Größenbeschränkungen erfolgte mit dem im Abschnitt 6.3.2 beschriebenen Algorithmus.

Der Graph 7.1 stellt die Entwicklung der Kosinus-Ähnlichkeit mit den verschiedenen Größenbeschränkungen dar. Wie zu erwarten steigt die durchschnittliche Ähnlichkeit mit der Anzahl der Klassen an. Einzig mit den strengen Größenbeschränkungen fiel die Ähnlichkeit bei der Erhöhung der Klassenanzahl von 1 000 auf 2 000 um 0,003.

Bei Vergleich der unterschiedlichen Größenbeschränkungen fällt auf, dass die weichen Größenbeschränkung nahezu keinen Einfluss auf die Qualität der Klassenzuordnungen haben. Bei zwei der fünf getesteten Größen für die Klassenanzahl konnten mit weichen Beschränkungen sogar geringfügig bessere Ergebnisse erzielt werden als mit dem K-Means Algorithmus ohne Größenbeschränkungen. Ein deutlicher Unterschied kann jedoch bei den strengen Größenbeschränkungen ausgemacht werden. Dabei nimmt die durchschnittliche Ähnlichkeit im Vergleich zu den konkurrierenden Ansätzen relativ um bis zu 6% ab. Dies ist vor allem damit zu erklären, dass aufgrund der geforderten Mindestgröße der Wortklassen nur noch 32 000 Wörter frei aufgeteilt werden konnten.

Bei den Clusteringverfahren ohne Größenbeschränkungen schwanken die Größen der verschiedenen Klassen dafür ziemlich stark. In Graph 7.2a werden die Klassengrößen der größten Klasse und der größten 10% der Klassen des K-Means Algorithmus mit den Obergrenzen der strengen und weichen Größenbeschränkungen verglichen. Dabei fällt auf, dass die Obergrenze der weichen Beschränkungen ziemlich nah an dem Durchschnitt der größten 10% der Klassen des K-Means Algorithmus liegt, bei dem Vergleich mit der Größe der jeweils größten Klasse stellt sich jedoch ein deutlicher Unterschied heraus.

Ein ähnliches Bild zeigt sich für die kleinsten erzeugten Klassen. Dabei besitzen die kleinsten Klassen, wenn der K-Means Algorithmus die Wörter in mindestens 2 000 Klassen aufteilt, nur 1 Wort. Die durchschnittliche Größe der untersten 10% der Klassen unterscheidet sich wiederum nur wenig von der geforderten Mindestgröße der

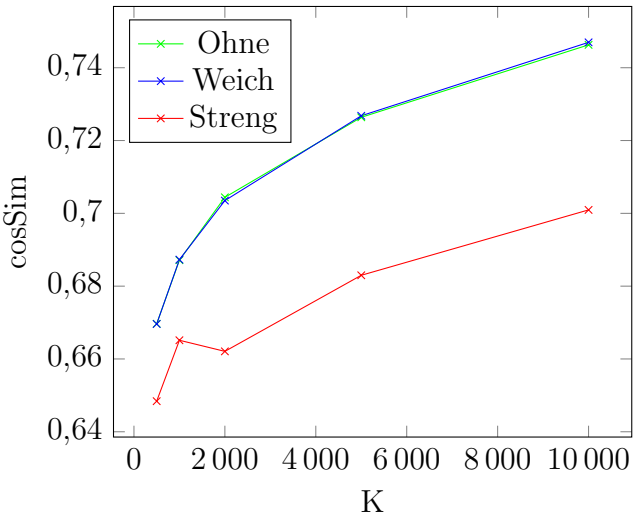


Abbildung 7.1: Vergleich der durchschnittlichen Kosinus-Ähnlichkeit (cosSim) der Klassenzuordnungen mit verschiedenen Größenbeschränkungen

schwachen Größenbeschränkungen, wobei ein deutlicher Unterschied zu den strengen Beschränkungen besteht.

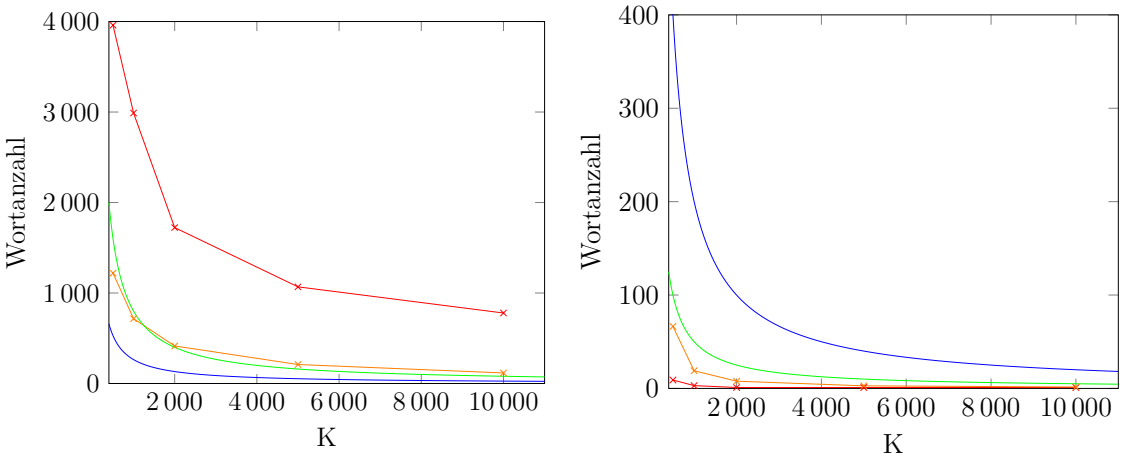
Die Größenbeschränkungen können im Hinblick auf die Auswertungen der Wahrscheinlichkeiten des neuronalen Netzes einen Unterschied machen. Bei der Berechnung der N-Gramm Wahrscheinlichkeit eines Wortes mithilfe der Klassenwahrscheinlichkeit des neuronalen Netzes und der Unigramm-Wahrscheinlichkeit des Wortes innerhalb der Klasse wird bei unterschiedlichen Klassengrößen die Bedeutung des Beitrags der beiden Komponenten verändert. Bei einer Klasse mit vielen Wörtern spielt die Unigramm-Wahrscheinlichkeit eine größere Rolle, da nach der Zuweisung einer Wahrscheinlichkeit an eine Klasse mit dem neuronalen Netz immer noch mithilfe der Unigramm-Wahrscheinlichkeit zwischen einer großen Anzahl an Wörtern unterschieden werden muss. Zudem ist es bei sehr kleinen Klassen wahrscheinlicher, dass in dem Trainingstext nur sehr wenige Beispiele der Wörter dieser Klasse vorkommen. Dies ist vor allem deswegen relevant, da nur die Wörter in Klassen unterteilt werden, die nicht zu den häufig vorkommenden Wörtern des Vokabulars zählen.

7.1.2 Unterklassen

In diesem Abschnitt wird die Zuordnung der Wörter zu ihrer Unterklasse bewertet. Dabei ist diese Zuordnung, wie in Absatz 6.3.2 beschrieben, von den davor erstellten Wortklassen abhängig. Die Anzahl der Unterklassen entspricht hierbei der Anzahl der Wörter der größten Klasse.

Auch für dieses sekundäre Clustering Verfahren werden zuerst einige Beispiele gezeigt. Bei den in Tabelle 7.3 gezeigten Beispielen wurden die Wörter in 1 000 Klassen unterteilt, die jeweils zwischen 100 und 1 000 Wörter beinhalten. Anschließend wurden die Wörter folglich auf 1 000 Unterklassen aufgeteilt.

Dabei wurde das Wort Europameisterschaften in eine Klasse mit vor allem zu internationalen Sportveranstaltungen in Bezug stehenden Wörtern wie Olympia, Goldmedaille, Geher und Synchronschwimmer eingeteilt. In der Unterklasse dieses Wortes finden sich immer noch Wörter, die im Bezug dem Thema Sport stehen, wie



(a) Obergrenzen

(b) Untergrenzen

Abbildung 7.2: Durchschnittliche Klassengrößen der kleinsten 10% und der größten 10% der Wortklassen (orange) und Größe der kleinsten und größten Klasse (rot) im Vergleich mit den Minimal- und Maximalgrößen der weichen (grün) und strengen (blau) Größenbeschränkungen

verschiedene Ländernamen (Paraguay, Montenegro) und die Wörter Viertelfinalbegegnungen, ersatzgeschwächten und Playoffs.

Dabei lässt sich im Wesentlichen jedoch erkennen, dass die Zuordnungen zu Unterklassen nicht an die Qualität der initialen Zuordnung in Klassen heranreicht. Beispielweise ist nicht mehr zu begründen, warum die Wörter Praktikanten, Lichterketten und Fußgängerzone in der selben Klasse sein sollten. Die schlechteren Ergebnisse im Vergleich zu den Zuordnungen in Klassen begründen sich jedoch durch die relativ strengen Bedingungen, die bei der Unterklassenzuordnung eingehalten werden müssen.

Dieser Eindruck spiegelt sich auch bei der durchschnittlichen Kosinus-Ähnlichkeit wieder. Dabei verdeutlicht sich jedoch eine klare Abhängigkeit von den verwendeten Größenbeschränkungen der Klassenzuordnung. Mit strengen Einschränkungen werden hierbei die schlechtesten Ergebnisse erzielt. Dies lässt sich vor allem durch die geringe Anzahl der Unterklassen erklären. Mit Abstand am Besten schneidet die Zuordnung in Unterklassen ab, die auf den Clusteringverfahren ohne Größenbeschränkungen basiert. Dabei werden die Wörter jedoch auch in bis zu 28 mal mehr Unterklassen aufgeteilt.

7.1.3 Fazit der Evaluation der Clusteringverfahren

Zusammenfassend lässt sich festhalten, dass die in eine Klasse unterteilten Wörter eine sehr starke Ähnlichkeit besitzen. Dies ist sowohl an den angeführten Beispielen als auch an der durchschnittlichen Kosinus-Ähnlichkeit erkennbar. Bei der Erstellung einer großen Anzahl an Klassen ohne und mit weichen Größenbeschränkungen konnten hierbei die besten Ergebnisse erzielt werden.

Die Unterteilung in Unterklassen ist dagegen stark von der Beschaffenheit der Klassen abhängig. Aufgrund der formulierten Bedingungen, die die Unterklassenzuordnung erfüllen muss, haben eine größere Anzahl an Klassen und stärkere Größenbeschränkungen der Klassen einen negativen Einfluss auf die Unterklassenzuordnung.

Europameisterschaften	WM Olympia Meisterschaften Goldmedaille Wettkämpfe Weltmeisterschaften Athletin Medaillengewinner Saisonhöhepunkt Wettkämpfer Olympiateilnahme Geher Synchronschwimmer
	ersatzgeschwächten Finninnen Halbschwergewicht individual Jahresvertrag Montenegro Olympiastadion Österreich Paraguay Playoffs qualifizierten Schnellster Tschechoslowakei Viertelfinalbegegnungen
Menschenansammlungen	Sicherheitskräfte Protesten Festnahmen Ausnahmezustand gewaltsam Zusammenstöße Regierungsgegner Plünderungen Staatsmacht protestierende Protestwellen bürgerkriegsartigen Meutereien
	Fußgängerzone Krach nächtliche Rechtsextremen Ferienzeit Zumutung Brennpunkten Zerstreuung Lärmbelästigungen Ärgernisse Bierzelten Sprechanlage Strafgesetze Drängelei Lichterketten Praktikanten

Tabelle 7.3: Wörter ausgewählter Klassen erstellt durch ein 1000-Means Clustering mit einer Minimalgröße von 100 Wörtern pro Klasse und einer Maximalgröße von 1 000 Wörtern pro Klasse

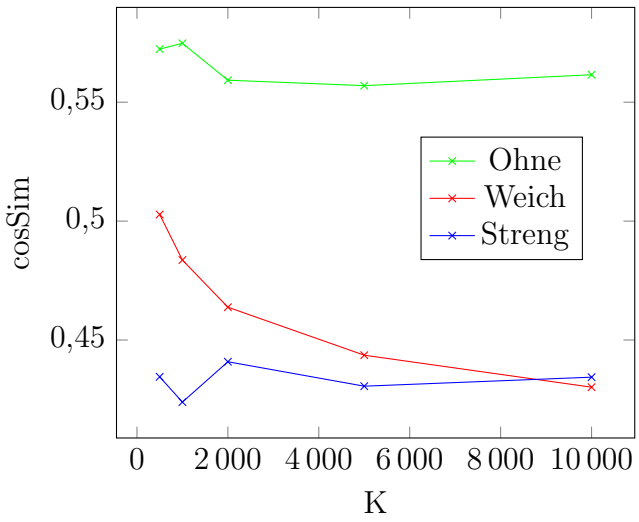


Abbildung 7.3: Vergleich der durchschnittlichen Kosinus-Ähnlichkeit (cosSim) der Unterklasse bei Veränderung der Größenbeschränkungen

Trotzdem lässt sich erkennen, dass immer noch viele anhand ihrer Bedeutung zusammengehörige Wörter der selben Unterklasse zugeordnet werden.

7.2 Das N-Gramm Sprachmodell und die Testdaten

In den folgenden Abschnitten werden nun die in Kapitel 4 eingeführten Konzepte evaluiert. Als Testdaten werden dafür Aufnahmen von verschiedenen Vorträgen mit den vorgestellten Sprachmodellen erkannt und ein Text wird ausgegeben, auf welchem dann eine Wortfehlerrate bestimmt wird.

Bei den für die Evaluation verwendeten Audioaufnahmen handelt es sich um TED[1] und TEDx Vorträge in deutscher Sprache. Diese Vorträge haben eine Länge von meist unter 20 Minuten und werden im Rahmen einer Konferenz gehalten. Die TEDx Konferenzen unterscheiden sich dabei lediglich dadurch, dass sie von unabhängigen Organisatoren veranstaltet werden.

Die Audioaufnahmen haben eine Länge von zwei Stunden und bestehen aus sieben Vorträgen von sieben unterschiedlichen Referenten. Der aus diesen Vorträgen erstellte Referenztext enthält dabei insgesamt 18 000 Wörter. Vorgeschlagen wurde dieser Datensatz im Jahr 2012 von dem *International Workshop on Spoken Language Translation*. Dieser internationale Arbeitskreis organisiert eine jährliche Evaluationskampagne, die ihren Fokus auf die automatische Übersetzung und auf die Erkennung gesprochener Sprache legt. Eine Zusammenfassung der Evaluationskampagne aus dem Jahr 2012 ist in [CBPS] zu finden.

Das Sprachmodell des neuronalen Netzes wird nicht als alleiniges Sprachmodell des Spracherkennungssystems verwendet, sondern es wird mit einem N-Gramm Sprachmodell interpoliert. Für dieses N-Gramm Sprachmodell wurde anhand der in Kapitel 5.4 vorgestellten Trainingsdaten ein Vokabular ausgewählt. Analog zu [KHMS$^+$] wurden dabei aus allen Trainingsdaten Unigramm-Sprachmodelle gebildet und daraus Unigramm-Wahrscheinlichkeiten für die einzelnen Wörter so erstellt, dass die Perplexität eines noch nicht gesehenen Referenztextes eines TED Vortrags minimiert wurde. Die 300 000 mit der höchsten Unigramm-Wahrscheinlichkeit gewichteten Wörter wurden dann als Vokabular ausgewählt. Zusammengesetzte Wörter wurden dabei wie in [KMHN$^+$] aufgeteilt. Um das Sprachmodell auf diesem Vokabular zu erstellen, wurde danach für alle verwendeten Trainingstexte jeweils ein eigenes Sprachmodell erstellt. Dafür wurde das SLIRM Toolkit [Stol$^+$02] mit dem modifizierten Kneser-Ney Smoothing verwendet. Die verschiedenen Sprachmodelle wurden dann linear interpoliert um wiederum die Perplexität auf einem noch nicht gesehenen Referenztext eines TED Vortrags zu minimieren.

Im Folgenden wird nun zunächst eine Referenzimplementierung des neuronalen Netzes vorgestellt, die nur einen Teil des gesamten Vokabulars betrachtet. Die beste aus dieser Implementierung resultierende Wortfehlerrate wird dann mit den neuen Ansätzen verglichen. Dabei wird der Fokus auf die Evaluation unterschiedlicher Strukturen der Ausgabeschicht des neuronalen Netzes und verschiedener Größenbeschränkungen für die Klassenzuordnungen gelegt. Des Weiteren wird der Einfluss der

[1] TED bezeichnet eine ursprünglich jährlich veranstaltete Innovations-Konferenz, deren Vorträge auf http://www.ted.com/ veröffentlicht werden

Größe der Trainingsdaten auf die Qualität des Spracherkennungssystems untersucht. Dafür werden die in Abschnitt 5.4 motivierten und in Tabelle 5.2 zusammengefassten Trainingstexte verwendet.

7.3 Referenzmodell des neuronalen Netzes

In diesem Abschnitt wird nun die Wortfehlerrate des Referenzmodells bestimmt, in dem nur die in der Shortlist enthaltenen Wörter mit dem Sprachmodell des neuronalen Netzes betrachtet werden. Dieses Sprachmodell wird dann mithilfe des Parameter α_s gewichtet und mit dem N-Gramm Sprachmodell interpoliert. Dabei wird das neuronale Netz mit dem Trainingstext t10 trainiert, der 33 Millionen Trainingsbeispiele für die Wörter der Shortlist enthält.

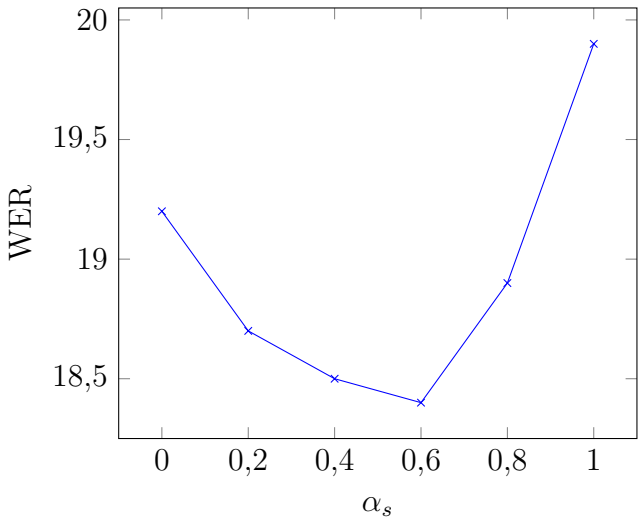


Abbildung 7.4: Entwicklung der Wortfehlerrate (WER) des Referenzmodell mit der Shortlist bei verschiedenen Interpolationsparametern α_s

Bei dieser Versuchsreihe wurde der Interpolationsparameter α_s jeweils um 0,2 erhöht. Für den Parameter $\alpha_s = 0$ wird somit nur das N-Gramm Sprachmodell evaluiert. Dieses erreicht eine Wortfehlerrate von 19,2%. Wie in Graph 7.4 zu erkennen ist, verbessert sich bei der Erhöhung des Parameters die Wortfehlerrate beständig bis zu einem Maximum von 18,4% bei $\alpha_s = 0{,}6$. Danach ist ein Abfall der Fehlerrate auf bis zu 19,9% zu beobachten, wenn die Wörter der Shortlist allein mit dem Sprachmodell des neuronalen Netzes betrachtet werden.

Der durch diesen Versuch gefundene, optimale Interpolationsparameter für das Sprachmodell deckt sich mit den Erkenntnissen aus anderen Arbeiten. Auch in [LOAG+13] wurden die besten Ergebnisse bei der Betrachtung einer Shortlist mit dem Sprachmodell des neuronalen Netzes mit Interpolationsgewichten zwischen 0,5 und 0,6 erreicht.

7.4 Neuronales Netz mit Klassen und Unigramm-Wahrscheinlichkeit

Nachdem für das Referenzmodell eine Wortfehlerrate auf dem Testdatensatz bestimmt wurde, wird nun der in Abschnitt 7.1.1 vorgestellte Ansatz, bei dem das

neuronale Netz zusätzlich die Wahrscheinlichkeit einer Klasse bestimmt, evaluiert. Die Architektur des hierfür verwendeten neuronalen Netzes ist in Abbildung 7.5 dargestellt. Dabei ist hervorzuheben, dass die Wahrscheinlichkeiten der Wörter der Shortlist $p(w|h)$ und die Wahrscheinlichkeiten einer Klasse $p(o|h)$ in einer Ausgabeschicht erlernt werden. Da mit dem Referenzmodell die besten Ergebnisse mit einem Interpolationsparameter von $\alpha_s = 0{,}6$ erreicht wurden, wird dieser Parameter weiterhin für die Wörter der Shortlist verwendet. Für die Interpolation der restlichen Wörter wird der Parameter α_r variiert, wobei er für die ersten Versuche auf 0,6 festgesetzt wird.

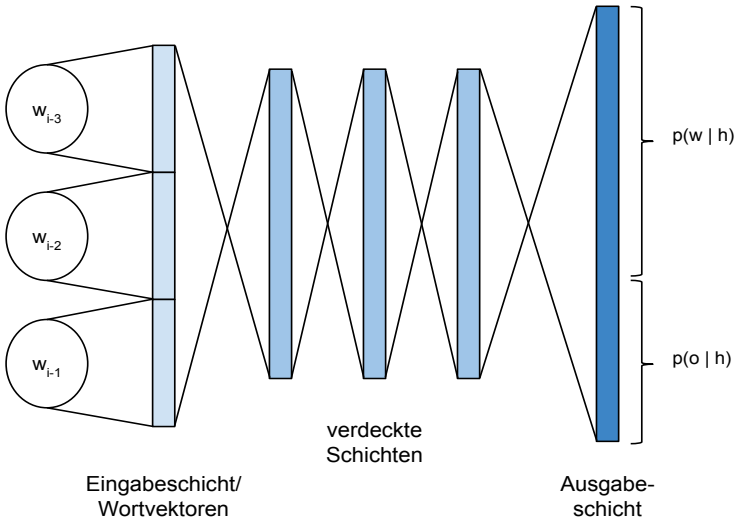


Abbildung 7.5: Architektur des neuronalen Netzes mit Klassen

Für das Schätzen der im Folgenden verwendeten Unigramm-Wahrscheinlichkeiten für das Auftreten eines Wortes gegeben seiner Klasse wurde ein deutlich größerer Trainingstext bestehend aus 1,6 Milliarden Wörtern verwendet. Um 0 Wahrscheinlichkeiten zu vermeiden, wurden 5% der Wahrscheinlichkeitsmasse gleichmäßig auf alle Wörter aufgeteilt. Die restlichen 95% dagegen wurden nach der Häufigkeit des Auftretens des jeweiligen Wortes im Trainingstext verteilt. Dieses Verfahren wurde als eine Smoothing Technik in Abschnitt 2.2 vorgestellt. Das neuronale Netz wurde, genau wie das Referenzmodell, mit den t10 Trainingsdaten trainiert. Dieser Trainingstext beinhaltet 31 Millionen Wörter der Shortlist und 2,4 Millionen der restlichen Wörter des Vokabulars.

7.4.1 Klassen ohne Größenbeschränkungen

Da dieser Ansatz auf davor erstellten Wortklassen basiert, werden im Folgenden verschiedene Klassenzuordnungen evaluiert. Bei einem ersten Testdurchlauf wurden die Wörter ohne Größenbeschränkungen mit dem K-Means Algorithmus in Klassen aufgeteilt. Dabei wurde zuerst eine Aufteilung in 500 Klassen betrachtet. Da bei dem Clustering-Verfahren knapp über 232 000 Wörter betrachtet werden, entspricht bei dieser Unterteilung die Anzahl der Klassen ungefähr der Quadratwurzel der Anzahl der Wörter. Bei gleich großen Klassen würde somit jede Klasse knapp 500 Wörter enthalten. Damit ist der Entscheidungsspielraum für die Wörter außerhalb der Shortlist ungefähr gleich zwischen dem neuronalen Netz und den Unigramm-Wahrscheinlichkeiten aufgeteilt. Bei dem beschriebenen Versuch mit 500 Klassen

ohne Größenbeschränkung wurde auf den Testdaten eine Wortfehlerrate von 18,7% gemessen.

Nun wurde der Ansatz auch mit anderen Klassengrößen getestet. Dabei wurde jeweils ein Sprachmodell mit 1 000, 2 000 und 5 000 Klassen untersucht. Die mit Abstand besten Ergebnisse konnten bei 1 000 verschiedenen Klassen erreicht werden. Dabei wurde eine Wortfehlerrate von 18,4% gemessen, die dem besten Ergebnis des Referenzmodells entspricht. Bei den übrigen Klassengrößen wurden, wie in dem blau gefärbten Graph in Abbildung 7.7 dargestellt, schlechtere Wortfehlerraten von 18,6% und 18,7% gemessen.

Für das beste Ergebnis einer Wortfehlerrate von 18,4%, die durch die Aufteilung in 1 000 Klassen ohne Größenbeschränkungen erreicht wurde, wurde eine Änderung des Parameters α_r evaluiert. Wie in Abbildung 7.6 dargestellt, wird bei einem höheren Wert des Parameters eine Verschlechterung der Fehlerrate auf 18,5% wahrgenommen. Jedoch konnte die Wortfehlerrate für $\alpha_r = 0{,}4$ auf 18,3% verbessert werden. Dies entspricht einem um 0,1% besseren Wert als das beste Ergebnis der Referenzimplementierung.

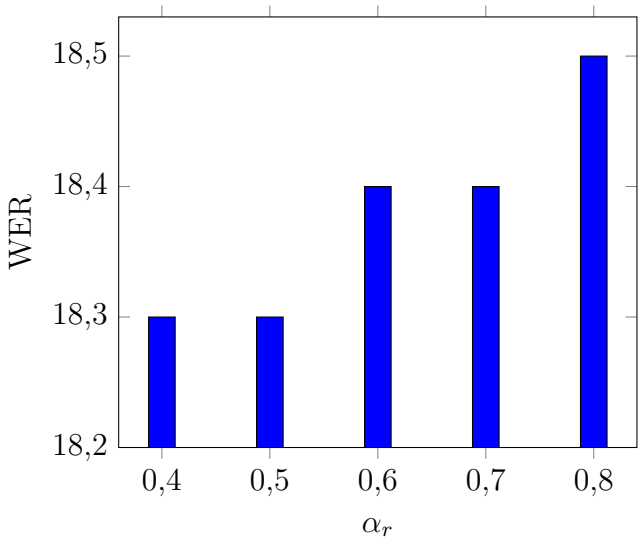


Abbildung 7.6: Entwicklung der Wortfehlerrate bei einem Clustering in 1 000 Klassen bei variablen α_r und festen $\alpha_s = 0{,}6$

7.4.2 Klassen mit Mindestgrößen

Da die Klassen ohne Größenbeschränkungen gebildet wurden, sind nun jedoch auch Klassen mit sehr wenigen Wörtern entstanden. Dabei ist es denkbar, dass bei einer sehr kleinen Klasse bestehend aus selten vorkommenden Wörtern keine oder nur sehr wenige Wörter dieser Klasse in dem Trainingstext vorkommen. Beispielsweise war bei einem Clustering in 2 000 Klassen das Wort „Hockenheimer“ das einzige Wort seiner Klasse und kam nur einmal in dem Trainingstext t10 vor. Um dies zu vermeiden, wurden auch Klassenzuordnungen mit einer Mindestgröße evaluiert. Bei einer Aufteilung der Wörter in K Klassen erhielt jede Klasse mindestens $\frac{200000}{K}$ Wörter, was auch der Mindestgröße der Klassenaufteilung mit strengen Größenbeschränkungen entspricht. Eine Maximalgröße wurde dabei nicht explizit gefordert, da eine große Klasse bei diesem Ansatz keine negativen Folgen für das Erlernen

des Sprachmodells hat. Bei einer großen Klasse erhält das neuronale Netzwerk nämlich viele Trainingsbeispiele und die Unigramm-Wahrscheinlichkeit lässt sich dabei weiterhin stabil ausrechnen, zumal diese auf einem deutlich größerem Trainingstext geschätzt worden ist.

Die mit diesen Klassen gemessenen Wortfehlerraten sind jedoch größtenteils schlechter als die ohne Größenbeschränkungen beobachteten Fehlerraten. Lediglich bei 500 erstellten Klassen konnte eine absolute Verbesserung von 0,1% erzielt werden. Mit den restlichen Klassengrößen verschlechterte sich die Wortfehlerrate jedoch um 0,1% bis 0,3%. In Abbildung 7.7 werden die Fehlerraten der beiden Clustering Methoden verglichen. Dabei ist vor allem die konstant schlechtere Wortfehlerrate bei den Klassen mit einer Minimalgröße zu erkennen.

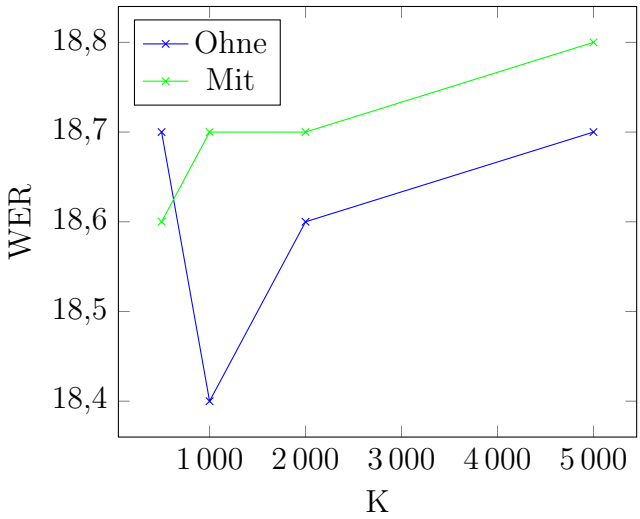


Abbildung 7.7: Entwicklung der Wortfehlerrate bei einem Clustering in K Klassen mit und ohne Größenbeschränkungen bei festen $\alpha_r = 0{,}6$ und $\alpha_s = 0{,}6$

Diese Verschlechterung könnte natürlich auf die deutlich schlechtere durchschnittliche Ähnlichkeit der Wörter zurückzuführen sein. Die durchschnittliche Kosinus-Ähnlichkeit ist aufgrund der Größenbeschränkungen um bis zu 0,031 geringer als bei dem Clusteringverfahren ohne Größenbeschränkung.

7.4.3 Wahl unabhängiger Interpolationsparameter

Um zu untersuchen, ob die Verschlechterungen für die übrigen Klassengrößen von den Wörtern verursacht worden sind, die nicht zur Shortlist gehören, wurde das Interpolationsgewicht α_r auf 0 gesetzt. Für diese Wörter wurde demnach die Wahrscheinlichkeit alleine durch das N-Gramm Sprachmodell bestimmt. Dabei wurden ähnliche Fehlerraten wie bei der vorherigen Versuchsreihe erzielt. Allein bei der Aufteilung in 1 000 Klassen wurde mit der Interpolation aller Wörter mit dem Faktor 0,6 eine Verschlechterung um 0,1% beobachtet. Bei der Aufteilung in 2 000 Klassen wurde dagegen eine Verbesserung um die gleiche Prozentzahl erreicht. Bei den restlichen Klassengrößen wurde jedoch keine Veränderung festgestellt. Diese Ergebnisse sind in Tabelle 7.4 dargestellt.

Damit zeigt sich, dass die Verschlechterungen im Vergleich zu dem Referenzmodell nicht direkt an dem Einfluss der zusätzlich betrachteten Wörter festzumachen sind.

K	$\alpha_r = 0$	$\alpha_r = 0{,}6$
500	18,7	18,7
1 000	18,4	18,4
2 000	18,6	18,5
5 000	18,7	18,8

Tabelle 7.4: Wortfehlerraten mit und ohne Benutzung der Wahrscheinlichkeiten des neuronalen Netzes für Wörter, die nicht in der Shortlist enthalten sind

Dagegen liegt die Vermutung nahe, dass die neu hinzugekommenen Neuronen für die Klassen während der Trainingsphase auch in einem signifikanten Maße die Gewichte des neuronalen Netzes beeinflussen, die für die Wörter der Shortlist relevant sind. Dadurch ließe sich die Änderung der Wortfehlerrate erklären.

7.5 Neuronales Netz mit Klassen und Unterklassen

Im Folgenden wird nun die Unigramm-Wahrscheinlichkeit durch die Wahrscheinlichkeit des Auftretens der Unterklasse eines Wortes ersetzt und damit der in Abschnitt 4.4.2 vorgestellte Ansatz evaluiert. In den Ausgabeschichten werden demnach neben den Wörtern der Shortlist sowohl die Klassen als auch die Unterklassen betrachtet. Dafür können entweder zwei oder drei Ausgabeschichten verwendet werden, je nachdem, ob die Wörter der Shortlist und die Klassen zusammen in einer Schicht behandelt werden. Der Interpolationsparameter α_s wird in den folgenden Experimenten auf 0,6 gesetzt und der Parameter α_r wird meist mit den Werten 0 und 0,6 belegt.

7.5.1 Zwei Ausgabeschichten

Für diesen Ansatz wurden mehrere Experimente mit verschiedenen Klassen- und Unterklassengrößen durchgeführt. Ein erster Versuch wurde mit recht strengen Größenbeschränkungen evaluiert, so dass sich die Anzahl der Klassen und die Anzahl der Unterklassen nicht stark unterscheiden. Dazu wurden die betrachteten Wörter in 500 Klassen aufgeteilt, so dass jede Klasse zwischen 400 und 600 Wörter enthält. Durch diese Aufteilung wurden dann die Wörter 600 verschiedenen Unterklassen zugeordnet. Wie in Abbildung 7.8 dargestellt, wird nun eine Ausgabeschicht für das Erlernen der Wahrscheinlichkeiten der Wörter der Shortlist und der Klassen verwendet und eine eigene Schicht für die Unterklassen. Die Wahrscheinlichkeiten der beiden Ausgabeschichten wurden in diesem Versuch mit dem t10 Text erlernt. Dabei wurde eine Wortfehlerrate von 18,6% gemessen. Wenn mit dem selben neuronalen Netz jedoch nur die Wörter der Shortlist betrachtet wurden, sank die Fehlerrate auf 18,5%. Demnach hat sich durch die Verwendung der Informationen über die erlernte Klassen- und Unterklassenzugehörigkeit die Qualität der Sprachmodells verschlechtert.

Dies kann unter anderem an den zwei folgenden Faktoren liegen. Die Menge der Trainingsbeispiele könnte für die Klassen und die Unterklassen nicht groß genug sein und die Anzahl der Klassen und der Unterklassen könnte zu strikt festgesetzt

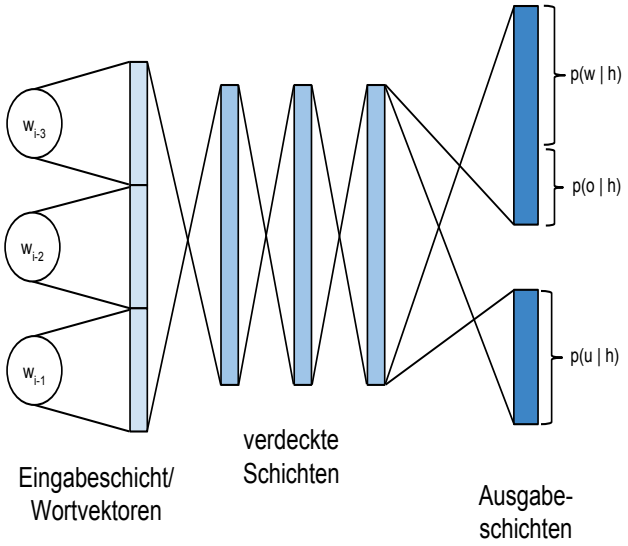


Abbildung 7.8: Architektur des neuronalen Netzes mit Klassen und Unterklassen bei zwei verwendeten Ausgabeschichten

worden sein, was eine qualitativ schlechtere Zuordnung zu den jeweiligen Klassen zur Folge hat. Deshalb wird nun die Ausgabeschicht der Unterklassen mit deutlich mehr Daten trainiert. Dafür wird der t50 Trainingstext verwendet, der 10 Millionen Beispiele für die in Unterklassen gegliederten Wörter enthält. Dadurch erhält die Schicht der Shortlist und der Klassen nur noch dreimal so viele Trainingsbeispiele. Des Weiteren wird zusätzlich eine Aufteilung der Wörter in 1 000 Klassen evaluiert, wobei dabei jede Klasse zwischen 100 und 1 000 Wörter enthält.

Bei der Aufteilung in 500 Klassen mit dem Parameter $\alpha_r = 0{,}6$ wurde eine Wortfehlerrate von 18,7% gemessen, wobei keine Veränderung durch ein Herabsetzen des Parameters auf 0 beziehungsweise auf 0,4 festgestellt wurde. Bei dem Sprachmodell mit 1 000 Klassen und 1 000 Unterklassen wurde ebenfalls eine Wortfehlerrate von 18,7% mit $\alpha_r = 0{,}6$ gemessen. Jedoch konnte dieses mal eine Änderung bei verschiedenen Interpolationsgewichten festgestellt werden. Mit dem Gewicht $\alpha_r = 0$ konnte zwar eine Wortfehlerrate von 18,6% festgestellt werden, bei einem Interpolationsgewicht von $\alpha_r = 0{,}4$ wurde aber eine weitere Verbesserung auf 18,5% gemessen. Dies entspricht zumindest einer Verbesserung der Fehlerrate um 0,1% im Vergleich zum Ignorieren der in Klassen aufgeteilten Wörter. Diese Ergebnisse sind im ersten Teil der Tabelle 7.5 zusammengefasst.

7.5.2 Drei Ausgabeschichten

Bei dem gerade betrachteten Ansatz mit zwei Schichten ist es nicht möglich, die Anzahl der Trainingsbeispiele für die Klassen zu erhöhen während die Beispiele für die Wörter der Shortlist konstant gehalten werden, da dadurch das Verhältnis zwischen den Klassen und den Wörtern der Shortlist verändert werden würde. Daher wird nun ein neuronales Netz mit drei verschiedenen Ausgabeschichten verwendet. Diesmal besitzt das neuronale Netz jeweils eine eigene Ausgabeschicht für die Wörter der Shortlist und für die Klassen. Diese Architektur ist in 7.9 dargestellt. Die Ausgabeschicht der Wörter der Shortlist wird nun weiterhin mit den t10 Trainingstext trainiert, wobei die beiden anderen Schichten mit den Trainingsbeispielen aus dem Text t50 trainiert werden.

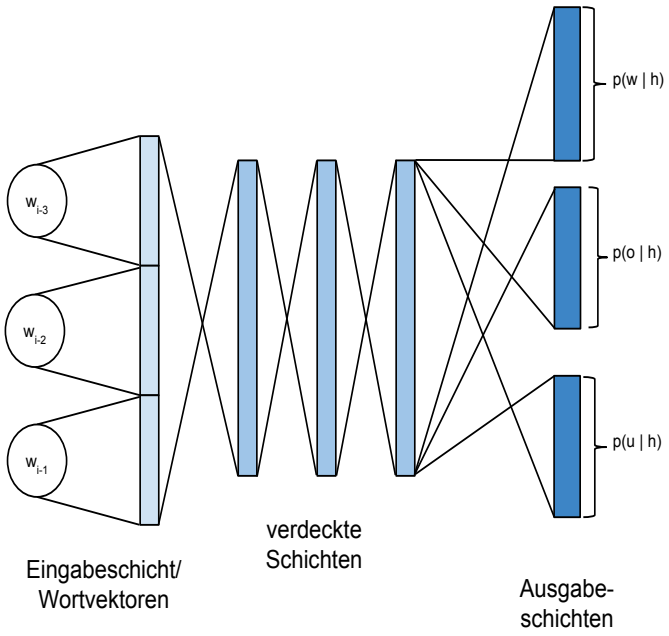


Abbildung 7.9: Architektur des neuronalen Netzes mit Klassen und Unterklassen bei drei verwendeten Ausgabeschichten

Mit diesem Ansatz wurde mit dem Sprachmodell, das auf 500 Klassen basiert, eine Wortfehlerrate von 18,8% gemessen, wobei ohne Verwendung der Wahrscheinlichkeiten der in Klassen unterteilten Wörter eine um 0,1% geringere Fehlerrate beobachtet wurde. Bei der Aufteilung in 1 000 Klassen konnte dagegen eine deutlich bessere Fehlerrate von 18,5% festgestellt werden, die auch im Vergleich zu der Fehlerrate, die mit dem Parameter $\alpha_r = 0$ gemessen wurde, eine Verbesserung von 0,1% darstellt.

In einer finalen Versuchsreihe mit diesen beiden Klassenunterteilungen wurde die Ausgabeschicht der Shortlist mit genauso vielen Beispielen wie die beiden anderen Schichten trainiert. Dabei wurde aber nicht mehr das in Absatz 5.4 erwähnte Verfahren zur Auswahl der Trainingssätze durch einen auf maschinelles Lernen basierenden Ansatz gewählt, da ein zu großer Anteil der Sätze der vorhandenen Texte ausgewählt werden musste. In dem Trainingstext *all* wurden deswegen die Sätze zufällig ausgewählt. In diesem Text sind nun auch 33 Millionen Beispiele für die Klassen und Unterklassen vorhanden, so dass die verschiedenen Schichten nun ungefähr gleich viele Trainingsdaten erhalten.

Mit der größeren Menge an Trainingsdaten konnten jedoch keine Verbesserungen erzielt werden. Mit 500 und 1 000 Klassen wurden jeweils Wortfehlerraten von 18,8% und 18,6% erzielt. Ohne die Betrachtung der Wörter mit dem neuronalen Netz, die nicht in der Shortlist enthalten sind, sank die Wortfehlerrate jeweils auf 18,6% und 18,5%. Daraus lässt sich schließen, dass allein durch eine größere Menge an Trainingsdaten keine Verbesserung der Fehlerraten möglich ist. Durch eine sinnvolle Auswahl der Trainingsbeispiele kann dagegen die Qualität des Sprachmodells gesteigert werden.

Die bis hierhin beschriebenen Experimente sind in Tabelle 7.5 zusammengefasst. Dabei lässt sich ein Unterschied zwischen den Wortfehlerraten mit den beiden verwendeten Klassengrößen erkennen. Bei der Zuordnung der Wörter in 500 verschie-

Schichten	Trainingsdaten	500 Klassen		1 000 Klassen	
		$\alpha_r = 0$	$\alpha_r = 0{,}6$	$\alpha_r = 0$	$\alpha_r = 0{,}6$
2	t10	18,5	18,6	-	-
	t50	18,7	18,7	18,6	18,7
3	t50	18,7	18,8	18,6	18,5
	all	18,6	18,8	18,5	18,6

Tabelle 7.5: Gemessene Wortfehlerraten bei Verwendung verschiedener Trainingstexte für die Klassen- und Unterklassenzugehörigkeiten. Die Ausgabeschicht, in der die Wörter der Shortlist enthalten sind, wurde dabei mit dem Text t10 trainiert.

dene Klassen konnte bei Hinzunahme der mit dem neuronalen Netz erlernten Wahrscheinlichkeiten der in Klassen unterteilten Wörter unabhängig von dem Trainingstext keine Verbesserung erzielt werden. Dahingegen konnte mit 1 000 Klassen bei Erhöhung des Parameter α_r von 0 auf 0,6 eine Verbesserung um 0,1% festgestellt werden, als eine Architektur mit 3 Ausgabeschichten benutzt wurde. Daher kann davon ausgegangen werden, dass zu strenge Größenbeschränkungen bei der Einteilung der Wörter in Klassen und Unterklassen sich kontraproduktiv auf die Qualität des Sprachmodells des neuronalen Netzes auswirken. Bei der Aufteilung der Wörter in 1 000 Klassen mit weicheren Größenbeschränkungen konnten somit auch bessere Ergebnisse erzielt werden.

Dies kann unter anderem an der deutlich besseren Kosinus-Ähnlichkeit bei der Aufteilung in 1 000 Klassen liegen. Die Kosinus-Ähnlichkeiten der beiden verwendeten Klassenzuordnungen sind in den ersten beiden Zeilen der Tabelle 7.6 dargestellt. Dabei hat die Aufteilung in 500 Klassen mit strengen Größenbeschränkungen im Vergleich zu den 1 000 Klassen, die zwischen 100 und 1 000 Wörter enthalten, eine um 0,044 schlechtere durchschnittliche Kosinus-Ähnlichkeit ihrer Klassen. Die durchschnittliche Ähnlichkeit der Unterklassen ist dabei im Vergleich sogar um 0,059 geringer.

7.5.3 Wahl verschiedener Größenbeschränkungen

Um zu untersuchen, ob mit anderen Größenbeschränkungen bessere Ergebnisse erzielt werden können, werden für die Aufteilung in 1 000 Klassen nun die in Abschnitt 7.1 eingeführten strengen Größenbeschränkungen getestet. Zudem wird ein Experiment mit 1 000 Klassen ohne Größenbeschränkungen durchgeführt. Diese beiden Zuordnungen sind auch in Tabelle 7.6 dargestellt.

Diese Klassenzuordnungen wurden mit beiden in den Abbildungen 7.9 und 7.8 dargestellten Architekturen des neuronalen Netzes mit zwei und drei Ausgabeschichten getestet. Dafür wurde für das Training der Ausgabeschichten, die nicht die Wörter der Shortlist enthalten, der Text t50 verwendet. Bei der Klassenzuordnung ohne Größenbeschränkungen wurde dabei mit beiden Architekturen des neuronalen Netzes eine Wortfehlerrate von 18,6% erzielt. Auch nach dem Setzen des Parameters α_r auf 0 wurde die selbe Wortfehlerrate gemessen. Bei der Klassenzuordnung mit strengen Größenbeschränkungen wurde sowohl bei Verwendung von zwei und drei Ausgabeschichten die unveränderte Wortfehlerrate von 18,6% erreicht. Auch wenn nur die Wörter der Shortlist mit dem neuronalen Netz betrachtet werden, wurde die selbe Fehlerrate erzielt.

Anzahl der Klassen	Min	Max	cosSim Klassen	cosSim Unterklassen
500	400	600	0,639	0,447
1 000	100	1 000	0,683	0,506
1 000	1	2 989	0,687	0,575
1 000	200	264	0,665	0,424

Tabelle 7.6: Die für die Sprachmodelle verwendeten Klassenzuordnungen mit den minimalen (Min) und maximalen (Max) Klassengrößen und der durchschnittlichen Kosinus-Ähnlichkeit (cosSim) ihrer Klassen

Größenbeschränkung	α_r	zwei Schichten	drei Schichten
weich	0,0	18,6	18,6
	0,6	18,7	18,5
streng	0,0	18,6	18,6
	0,6	18,6	18,6
ohne	0,0	18,6	18,6
	0,6	18,6	18,6

Tabelle 7.7: Gemessene Wortfehlerraten bei Verwendung verschiedener Größenbeschränkung bei 1 000 verschiedenen Klassen.

In Tabelle 7.7 sind die Fehlerraten, die mit den verschiedenen Größenbeschränkungen bei 1 000 erstellten Klassen gemessen wurden, zusammengefasst. Dabei zeigt sich, dass auch hier die besten Ergebnisse mit den weichen Größenbeschränkungen und drei Ausgabeschichten erzielt werden konnten. Der Vorteil der drei Ausgabeschichten ist es, dass dadurch sowohl die Wahrscheinlichkeit der Klassen als auch der Unterklassen mit einem größeren Trainingstext als die Schicht mit den Wörtern der Shortlist trainiert werden können. Dies macht sich vor allem bei den weichen Größenbeschränkungen bemerkbar, bei denen die Wortfehlerrate bei Hinzunahme einer eigenen Schicht für die Klassen um 0,2% gesteigert werden konnte.

7.6 Erhöhung der Anzahl der Neuronen pro verdeckter Schicht

Im Vergleich zu dem Referenzmodell wurden in allen Experimenten, bei denen mehrere Ausgabeschichten verwendet worden sind, schlechtere Ergebnisse erzielt. Auch bei der in Abschnitt 7.4 evaluierten Architektur mit nur einer Ausgabeschicht wurden bis auf eine Ausnahme schlechtere Wortfehlerraten gemessen. Diese Beobachtung konnte auch mit dem Interpolationsparameter $\alpha_r = 0$ gemacht werden. Das bedeutet, dass sich durch die Hinzunahme der Klassen und Unterklassen in das Sprachmodell des neuronalen Netzes die erlernten Wahrscheinlichkeiten der Wörter der Shortlist verschlechtern.

Dies kann daran liegen, dass die verdeckten Schichten aus zu wenigen Neuronen bestehen um die höhere Anzahl an Ausgabewahrscheinlichkeiten darzustellen. Daher wird im Folgenden die Anzahl der Neuronen der verdeckten Schichten von 600 auf 800 erhöht. Mit 1 000 und 2 000 Klassen wurde damit der Ansatz mit den Unigramm-Wahrscheinlichkeiten evaluiert. Für den Ansatz mit den Klassen und Unterklassen

	600 Neuronen	800 Neuronen
1 000 Klassen	18,4	18,6
2 000 Klassen	18,6	18,6
2 Schichten	18,6	18,7
3 Schichten	18,6	18,6

Tabelle 7.8: Wortfehlerraten mit 600 und 800 Neuronen pro verdeckter Schicht im Vergleich. Getestet wurde das neuronale Netz mit Klassen und Unigramm-Wahrscheinlichkeiten und das neuronale Netz mit Klassen und Unterklassen mit einer unterschiedlichen Anzahl an Schichten.

	Absolut	prozentual
Shortlist	17 277	94,2%
Vokabular	18 195	99,2%
OOV	148	0,8%
Gesamt	18 343	100%

Tabelle 7.9: Verteilung der Wörter des Referenztextes

wurde die Klassenzuordnung mit 1 000 Klassen bei weichen Größenbeschränkungen mit zwei und mit drei Ausgabeschichten getestet.

Die dabei beobachteten Wortfehlerraten sind in Tabelle 7.8 dargestellt. Diese Ergebnisse wurden mit den Parametern $\alpha_s = 0{,}6$ und $\alpha_r = 0$ gemessen, um den Einfluss der verschiedenen Modelle allein auf die Wörter der Shortlist zu erkennen. Bei Erhöhung der Anzahl der Neuronen der verdeckten Schichten konnte dabei jedoch keine Verbesserung erkannt werden. Unabhängig von der Architektur des neuronalen Netzes verschlechterte sich die Wortfehlerrate oder blieb konstant. Demnach konnte durch diese Erhöhung der Anzahl der Neuronen die beobachtete Veränderung der Wahrscheinlichkeiten der Wörter der Shortlist nicht verhindert werden.

7.7 Fehleranalyse

In diesem Abschnitt werden nun die erzielten Ergebnisse genauer begutachtet. Dazu wird zuerst die Aufteilung der Wörter des Referenztextes analysiert und danach werden die Fehler betrachtet, die durch das Spracherkennungssystem gemacht wurden.

Von den über 18 000 Wörtern des Referenztextes waren nur 5% der Wörter in dem Vokabular und gleichzeitig nicht in der Shortlist enthalten, wohingegen 94,2% der Wörter des Textes Teil der Shortlist waren. Lediglich 0,8% der Wörter waren demnach nicht im Vokabular enthalten (OOV, englisch: Out Of Vocabulary). Zusammengesetzte Wörter des Referenztextes wurden für die Berechnung dieser Wahrscheinlichkeiten aufgeteilt. Das neu erstellte Sprachmodell ändert somit nur die Wahrscheinlichkeit von 918 der in der Aufnahme vorkommenden Wörter. Die genaue Verteilung der Wörter des Referenztextes ist Tabelle 7.9 zu entnehmen. Durch das Referenzmodell, welches die Wörter der Shortlist betrachtet, konnte die Wortfehlerrate dabei um 0,8% verbessert werden. Bei Annahme eines linearen Zusammenhangs zwischen der Abdeckung der mit dem neuronalen Netz betrachteten Wörter auf den

Art des Fehlers	$\alpha_r = 0$	$\alpha_r = 0{,}4$	$\alpha_r = 0{,}6$
Shortlist	1 663	1 670	1677
restliches Vokabular	209	205	208

Tabelle 7.10: Verteilung der nicht erkannten Wörter des Referenztextes bei einem Sprachmodell, das auf 1 000 verschiedenen Klassen basiert

Testaufnahmen und der Wortfehlerrate würde dies nur einer weiteren Verbesserung um $0{,}8\% \cdot \frac{5{,}0}{94{,}2} = 0{,}042\%$ entsprechen. Deshalb erhält die absolute Verbesserung der Wortfehlerrate von 0,1% durch das Sprachmodell des neuronalen Netzes mit den zusätzlichen Klassenwahrscheinlichkeiten eine größere Relevanz.

Nun werden die Fehler analysiert, die der Spracherkenner gemacht hat. Dafür wird die Anzahl der Wörter des Referenztextes betrachtet, die nicht richtig erkannt worden sind. Diese nicht erkannten Wörter werden nach ihrer Zugehörigkeit in Wörter der Shortlist und des restlichen Vokabulars aufgeteilt. Die Fehler wurden dabei in dem Spracherkennungssystem auf den Referenztext zurückgeführt, in dem zusammengesetzte Wörter noch nicht aufgeteilt wurden. Die zusammengesetzten Wörter werden dem restlichen Vokabular zugeordnet, falls sie ein Teilwort enthalten, das nicht in der Shortlist enthalten ist, aber Teil des Vokabulars ist. Wenn alle Teilwörter in der Shortlist enthalten sind, wird das Wort der Shortlist zugeordnet.

Dabei ist die Verteilung der Fehler in Tabelle 7.10 für ein Sprachmodell mit 1 000 verwendeten Klassen dargestellt. Somit fielen über 88% der falsch erkannten Wörter des Vokabulars auf die Wörter der Shortlist ab und nur circa 11% auf die Wörter des restlichen Vokabulars. Mit dem optimalen Interpolationsparameter $\alpha_r = 0{,}4$, mit dem die Wortfehlerrate von 18,3% erreicht wurde, konnte dabei auch die geringste Anzahl an falsch erkannten Wörtern des restlichen Vokabulars festgestellt werden. Jedoch fiel die Anzahl der falsch erkannten Wörter für den Parameter $\alpha_r = 0$, mit dem folglich ausschließlich das N-Gramm Sprachmodell für die restlichen Wörter verwendet wurde, nur um vier Fehler höher aus.

Ein Beispiel, das von dem klassenbasierten Sprachmodell falsch und von dem Referenzmodell richtig erkannt wurde, ist das Wort „hineinzudenken“. Dieses Wort war das einzige Wort seiner Klasse und kam in dem verwendeten Trainingstext nur in einem Satz vor. Dieser Satz ist zwar mehrere Male in dem Trainingstext vorhanden, das Wort wurde von dem neuronalen Netz allerdings trotzdem nur in einem Kontext gesehen. Damit kann erklärt werden, warum bei den klassenbasierten Ansatz bei diesem Wort ein Fehler aufgetreten ist.

Das Wort „firmiert“ ist dagegen ein gutes Beispiel für ein Wort, welches mit dem klassenbasierten Sprachmodell richtig und mit dem Referenzmodell falsch erkannt wurde. Dieses Wort ist von dem Clusteringalgorithmus in eine kleine Klasse mit sechs anderen Wörtern aufgeteilt worden. Darunter waren die Wörter „firmierte“, „firmieren“, „umfirmiert“ und „konfirmiert“. Dabei kam jedes dieser Wörter auch häufiger in dem Trainingstext vor. Als Grund für die Vermeidung des Fehlers könnte demnach die wenige und sehr ähnliche Wörter beinhaltende Wortklasse genannt werden. Bei dem Training des neuronalen Netzes wurden dadurch die Beispiele in mehr Kontexten gesehen, als wenn nur das einzelne Wort „firmiert“ betrachtet worden wäre.

Natürlich zeigen diese geringen Fehlerzahlen auch, dass die Testdaten nicht ausreichend groß sind um eine repräsentative Analyse der verschiedenen klassenbasierten Sprachmodelle zu ermöglichen. Dies zeigt vor allem die Tatsache, dass in dem Referenztext der Aufnahme nur weniger als 700 Wörter des Vokabulars vorkommen, die nicht auch in der Shortlist enthalten sind. Dagegen beinhaltet das Vokabular deutlich über 200 000 Wörter, welche nicht der Shortlist angehören. Dadurch ist nur ein sehr geringer Anteil der in Klassen unterteilten Wörter wirklich relevant für die Erkennung dieser Audiosequenz.

7.8 Zusammenfassung

In diesem Kapitel wurden verschiedene Experimente mit aus neuronalen Netzen erstellten Sprachmodellen durchgeführt. Diese Sprachmodelle wurden jeweils mit einem N-Gramm Sprachmodell interpoliert, welches auf dem Datensatz eine Wortfehlerrate von 19,2% aufweist. Für ein Referenzmodell, das mit dem neuronalen Netz nur knapp über 18 000 häufig vorkommende Wörter des Vokabulars betrachtet, wurde das Interpolationsgewicht mit dem N-Gramm Sprachmodell optimiert. Dabei wurde eine absolute Verbesserung der Wortfehlerrate um bis zu 0,8% auf 18,4% erreicht.

In einem neuem Ansatz wurden mit dem Sprachmodell des neuronalen Netzes alle Wörter des Vokabulars betrachtet. Dafür wurden die restlichen Wörter des Vokabulars in Klassen aufgeteilt, deren Wahrscheinlichkeit durch das neuronale Netz erlernt wurde. Über die Unigramm-Wahrscheinlichkeit eines Wortes gegeben seiner Klasse und der erlernten Wahrscheinlichkeit des Auftretens der Klasse wurde in diesem Modell die Wahrscheinlichkeit eines Wortes bestimmt. Hierbei wurden Sprachmodelle mit Klassen mit strengen Größenbeschränkungen und Klassen ohne Größenbeschränkungen erstellt und evaluiert. Dabei wurden mit Sprachmodellen, die auf den Klassenzuordnungen ohne Größenzuordnungen basieren, konstant bessere Ergebnisse als mit den Sprachmodellen mit Größenbeschränkungen erzielt. Die beste mit diesem Ansatz erreichte Wortfehlerrate von 18,3% konnte mit 1 000 Klassen gemessen werden, was einer weiteren Verbesserung um 0,1% im Vergleich zu dem besten Wert des Referenzmodells entspricht.

Außerdem wurden in einem weiteren Ansatz die davor verwendeten Unigramm-Wahrscheinlichkeiten durch die Zugehörigkeit der Wörter zu einer Unterklasse ersetzt. Dabei wurden Architekturen des neuronalen Netzes mit zwei und drei verschiedenen Ausgabeschichten evaluiert. Zudem wurden die Sprachmodelle mit einer verschiedenen Anzahl an Klassen und mit unterschiedlichen Klassengrößen getestet. Dabei wurden die besten Ergebnisse bei einer Unterteilung der Wörter in 1 000 Klassen und weichen Größenbeschränkungen erzielt.

Die Wortfehlerraten dieses Ansatzes blieben aber über denen des Referenzmodells. Auch bei ausschließlicher Betrachtung der Wahrscheinlichkeiten der Wörter der Shortlist mit demselben neuronalen Netz verschlechterte sich die Wortfehlerrate bei diesem Ansatz. Mit der Hinzunahme der Wahrscheinlichkeiten der restlichen Wörter konnte lediglich die Wortfehlerrate gegenüber der ausschließlichen Betrachtung der Shortlist um 0,1% verringert werden. Um festzustellen, ob durch die Anzahl der Neuronen der verdeckten Schicht die Anzahl der zu erlernenden Ausgabewahrscheinlichkeiten des neuronalen Netzes beschränkt werden und die Verschlechterungen aus

diesem Grund auf die zusätzliche Betrachtung der Klassen und Unterklassen zurückzuführen ist, wurde die verdeckte Schicht vergrößert. Jedoch konnte dadurch keine Verbesserung der Wortfehlerrate erreicht werden.

8. Zusammenfassung und Ausblick

In dieser Arbeit werden klassenbasierte Sprachmodelle mit neuronalen Netzen erforscht, in ein Spracherkennungssystem integriert und anhand einer Reihe von Vorträgen evaluiert. Bei der Betrachtung des gesamten Vokabulars einer gesprochenen Sprache in der Ausgabeschicht eines neuronalen Netzes ist es derzeit mit einem Spracherkennungssystem meist nicht möglich Audioaufnahmen in einer angemessen Zeit zu erkennen. Deswegen wird zuerst eine Referenzimplementierung getestet, die mit dem neuronalen Netz lediglich eine Teilmenge des Vokabulars betrachtet. Im Vergleich zu einem N-Gramm basierten Sprachmodell kann dabei eine absolute Verbesserung der Wortfehlerrate um 0,8% auf 18,4% beobachtet werden.

Um die Größe der Ausgabeschicht des neuronalen Netzes zu beschränken, werden in den klassenbasierten Sprachmodellen die restlichen Wörter des Vokabulars auf Basis vorher erlernter Merkmalsvektoren in Klassen aufgeteilt, die nun auch von dem neuronalen Netz betrachtet werden. Die Wahrscheinlichkeit dieser Wörter wurde dann mithilfe ihrer Klassenzugehörigkeit und einer Unigramm-Wahrscheinlichkeit, die die Häufigkeit des Auftretens eines Wortes in seiner Klasse darstellt, bestimmt. Dieses Modell wurde mit einer unterschiedlichen Anzahl an Klassen getestet. Dafür wurden die 230 000 Wörter des Vokabulars, die nicht einzeln in der Ausgabeschicht betrachtet werden, in 500 bis 5 000 verschiedene Klassen unterteilt. Zudem wurden verschiedene Parameter, die für die Interpolation zwischen den Wahrscheinlichkeiten des Sprachmodells des neuronalen Netzes und dem N-Gramm Sprachmodell verantwortlich sind, getestet. Mit einer Einteilung der Wörter in 1 000 Klassen konnte dabei eine Steigerung der Wortfehlerrate um weitere 0,1% auf 18,3% beobachtet werden. Bei den anderen Klassengrößen wurde dagegen lediglich eine Fehlerrate zwischen 18,6% und 18,7% festgestellt. Zusätzlich wurde überprüft, ob Größenbeschränkungen für die Klassen eine Verbesserung für das Sprachmodell bewirken. Durch diese Restriktion konnte jedoch keine weitere Verbesserung erreicht werden.

In einem anderen Ansatz wurde die Unigramm-Wahrscheinlichkeit durch die Zugehörigkeit der Wörter zu einer Unterklasse ersetzt, die auch von dem neuronalen Netz betrachtet wurde. Um dieses Modell umzusetzen, wurden die Wörter so in Unterklassen aufgeteilt, dass keine zwei Wörter sowohl der selben Klasse als auch der selben Unterklasse angehören. Dabei wurden sowohl eine Architektur des neuronalen

Netzes evaluiert in dem die einzeln betrachteten Wörter und die Klassen gemeinsam in einer Ausgabeschicht betrachtet wurden, als auch ein Modell bei dem sie jeweils mit einer eigenen Ausgabeschicht modelliert wurden. Die Unterklassen wurden dabei immer in einer separaten Schicht betrachtet. Dabei konnte bei der Architektur mit drei Schichten ein leicht besseres Ergebnis einer Wortfehlerrate von 18,5% erreicht werden als bei zwei verwendeten Schichten.

Zudem wurden unterschiedlich strenge Größenbeschränkungen für die Klassenzuordnungen und verschiedene Interpolationsparameter evaluiert. Dabei wurde festgestellt, dass mit einer Aufteilung der Wörter in 1 000 Klassen und Unterklassen, bei der jede Klasse zwischen 100 und 1 000 Wörter enthält, bessere Ergebnisse erzielt werden konnten als mit strengeren oder ohne Größenbeschränkungen. Mit den verschiedenen Interpolationsparametern für die einzeln betrachteten Wörter und die restlichen Wörter konnte die Erkenntnis gewonnen werden, dass sich die Wahrscheinlichkeit der einzeln betrachteten Wörter im Gegensatz zum Referenzmodell veränderte und zu einer schlechteren Fehlerrate führte, als zusätzlich Klassen und Unterklassen mit dem neuronalen Netz betrachtet wurden. Dagegen konnte bei separater Betrachtung der restlichen Wörter eine leichte Verbesserung der Wortfehlerrate gemessen werden.

Aufgrund der hohen Abdeckung der Teilmenge des Vokabulars, die mit dem neuronalen Netz einzeln betrachtet wurde, stellt die Verbesserung der Wortfehlerrate um 0,1% im Vergleich zu der erzielten Verbesserung des Referenzmodell bereits eine beachtliche Verbesserung dar.

Um die in dieser Arbeit vorgestellten Konzepte weiter zu verbessern, könnte die Unigramm-Wahrscheinlichkeit durch eine Wahrscheinlichkeit, die einen längeren Kontext der davor aufgetretenen Wörter betrachtet, ersetzt werden. Die Aufteilung der Wörter in Klassen könnte zudem nicht nur mehr mithilfe der Merkmalsvektoren vorgenommen werden. Eine weitere Möglichkeit wäre zum Beispiel die Verwendung der Transinformation, die in [BdMP$^+$92] zur Bildung von Klassen verwendet wurde. Zudem kann es vor allem bei der zusätzlichen Aufteilung der Wörter in Unterklassen Sinn machen, auch morphologische Merkmale der Wörter in die Bildung der Klassenzuordnungen miteinzubeziehen.

Außerdem sind weitere Tests dieser Sprachmodelle mit einer größeren Menge an Audioaufnahmen und in anderen Bereichen sinnvoll. Vor allem in Bereichen, in denen eine große Menge an Fachwörtern benutzt wird, könnten diese in das Vokabular aufgenommen werden und mit Hilfe der klassenbasierten Sprachmodelle effizient mit dem neuronalen Netz betrachtet werden. Auch bei Sprachen, für die eine kleinere Menge an Trainingsdaten vorhanden ist und bei denen ein N-Gramm Sprachmodell daher schlechtere Ergebnisse liefert, sind mit diesem Ansatz größere Verbesserungen zu erwarten.

Literaturverzeichnis

[BaGh] A. Banerjee und J. Ghosh. Scalable Clustering Algorithms with Balancing Constraints. Technischer Bericht.

[BaMC05] M. Balakrishna, D. Moldovan und E. K. Cave. Automatic creation and tuning of context free grammars for interactive voice response systems. In *Natural Language Processing and Knowledge Engineering, 2005. IEEE NLP-KE'05. Proceedings of 2005 IEEE International Conference on*. IEEE, 2005, S. 158–163.

[Bazz02] I. Bazzi. *Modelling out-of-vocabulary words for robust speech recognition*. Dissertation, Massachusetts Institute of Technology, 2002.

[BBBL+10] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley und Y. Bengio. Theano: a CPU and GPU Math Expression Compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, Juni 2010. Oral Presentation.

[BdMP+92] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra und J. C. Lai. Class-based N-gram Models of Natural Language. *Comput. Linguist.* 18(4), Dezember 1992, S. 467–479.

[BDVJ03] Y. Bengio, R. Ducharme, P. Vincent und C. Janvin. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* Band 3, 2003, S. 1137–1155.

[CBPS] M. F. M. Cettolo, L. Bentivogli, M. Paul und S. Stüker. Overview of the IWSLT 2012 Evaluation Campaign.

[CCCJ13] B. Chen, Y.-W. Chen, K.-Y. Chen und E.-E. Jan. Effective pseudo-relevance feedback for language modeling in speech recognition. In *ASRU*. IEEE, 2013, S. 13–18.

[CCGG98] S. F. Chen, S. F. Chen, J. Goodman und J. Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. Technischer Bericht, 1998.

[ChGo96] S. F. Chen und J. Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics, S. 310–318.

[DhMo01] I. S. Dhillon und D. S. Modha. Concept Decompositions for Large Sparse Text Data Using Clustering. *Mach. Learn.* 42(1-2), Januar 2001, S. 143–175.

[Du14] K.-L. Du. Neural Networks and Statistical Learning, 2014.

[DZHL+14] J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. Schwartz und J. Makhoul. Fast and Robust Neural Network Joint Models for Statistical Machine Translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Baltimore, Maryland, June 2014. Association for Computational Linguistics, S. 1370–1380.

[FuWK07] C. Fuegen, A. Waibel und M. Kolss. Simultaneous translation of lectures and speeches. *Machine Translation* 21(4), 2007, S. 209–252.

[GKKC07] F. Grezl, M. Karafiát, S. Kontár und J. Cernocky. Probabilistic and bottle-neck features for LVCSR of meetings. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, Band 4. IEEE, 2007, S. IV–757.

[Good01] J. T. Goodman. A bit of progress in language modeling. *Computer Speech and Language* 15(4), 2001, S. 403 – 434.

[HoSW89] K. Hornik, M. Stinchcombe und H. White. Multilayer feedforward networks are universal approximators. *Neural networks* 2(5), 1989, S. 359–366.

[Jeli97] F. Jelinek. *Statistical methods for speech recognition*. Language, speech and communicationA Bradford book. MIT Press, Cambridge, Mass. [u.a.]. 1997.

[KHMS+] K. Kilgour, M. Heck, M. Müller, M. Sperber, S. Stüker und A. Waibel. The 2014 KIT IWSLT Speech-to-Text Systems for English, German and Italian.

[KMHN+] K. Kilgour, C. Mohr, M. Heck, Q. B. Nguyen, V. H. Nguyen, E. Shin, I. Tseyzer, J. Gehring, M. Müller, M. Sperber und andere. The 2013 KIT IWSLT Speech-to-Text Systems for German and English.

[KnNe95] R. Kneser und H. Ney. Improved Backing-off for M-gram Language Modeling. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Mai 1995, S. 181–184.

[Le12] H. S. Le. *Continuous space models with neural networks in natural language processing*. Theses, Université Paris Sud - Paris XI, Dezember 2012.

[LOAG+13] H. S. Le, I. Oparin, A. Allauzen, J.-L. Gauvain und F. Yvon. Structured output layer neural network language models for speech recognition. *IEEE Transactions on Audio, Speech and Language Processing* 21(1), 2013, S. 197–206.

[LOMA+11] H. S. Le, I. Oparin, A. Messaoudi, A. Allauzen, J.-L. Gauvain und F. Yvon. Large vocabulary SOUL neural network language models. In *Annual Conference of the International Speech Communication Association (INTERSPEECH 2011)*, Florence, Italy, 27/08 au 31/08 2011. S. 1469–1472.

[MCCD13a] T. Mikolov, K. Chen, G. Corrado und J. Dean. Efficient Estimation of Word Representations in Vector Space. *CoRR* Band abs/1301.3781, 2013.

[MCCD13b] T. Mikolov, K. Chen, G. Corrado und J. Dean. Efficient Estimation of Word Representations in Vector Space. *CoRR* Band abs/1301.3781, 2013.

[MDKB+11] T. Mikolov, A. Deoras, S. Kombrink, L. Burget und J. H. Cernocky. Empirical Evaluation and Combination of Advanced Language Modeling Techniques. In *Interspeech*. ISCA, August 2011.

[Mitc97] T. M. Mitchell. *Machine learning*. McGraw-Hill Series in Computer ScienceMcGraw-Hill international editions. McGraw-Hill, New York [u.a.]. Internat. ed.. Auflage, 1997.

[MKBC+10] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký und S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, 2010, S. 1045–1048.

[NiWW98] T. Niesler, E. W. D. Whittaker und P. Woodland. Comparison Of Part-Of-Speech And Automatically Derived Category-Based Language Models For Speech Recognition. In *Proc. ICASSPÕ98*, 1998, S. 177–180.

[Schu95] E. G. Schukat-Talamazzini. *Automatische Spracherkennung - Grundlagen, statistische Modelle und effiziente Algorithmen*. Künstliche Intelligenz. Vieweg. 1995.

[Schw07] H. Schwenk. Continuous Space Language Models. *Comput. Speech Lang.* 21(3), Juli 2007, S. 492–518.

[Stol+02] A. Stolcke und andere. SRILM-an extensible language modeling toolkit. In *INTERSPEECH*, 2002.

[Tick13] J. L. Ticknor. A Bayesian regularized artificial neural network for stock market forecasting. *Expert Systems with Applications* 40(14), 2013, S. 5501 – 5506.

[VLBM08] P. Vincent, H. Larochelle, Y. Bengio und P.-A. Manzagol. Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, New York, NY, USA, 2008. ACM, S. 1096–1103.

[VTBE14] O. Vinyals, A. Toshev, S. Bengio und D. Erhan. Show and Tell: A Neural Image Caption Generator. *CoRR* Band abs/1411.4555, 2014.

[WFHS+14] C. Wu, W. Fan, Y. He, J. Sun und S. Naoi. Handwritten Character Recognition by Alternately Trained Relaxation Convolutional Neural Network. *Submitted to ICFHR*, 2014.

[XuFS14] B. Xue, C. Fu und Z. Shaobin. A Study on Sentiment Computing and Classification of Sina Weibo with Word2vec. In *Big Data (BigData Congress), 2014 IEEE International Congress on*. IEEE, 2014, S. 358–363.