Karlsruhe Institute of Technology

Themenspezifische Vor-Adaptierung von Sprachmodellen

Studienarbeit
von

Ying Xu

An der Fakultät für Informatik
Institut für Anthropomatik (IFA)

Erstgutachter:	Prof. Dr. Alex Waibel
Betreuender Mitarbeiter:	Kevin Kilgour

Bearbeitungszeit: 01. Dezember 2010 – 28. Februar 2011

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

Karlsruhe, den 28. Februar 2011

Abstrakt

Ein gutes Sprachmodell spielt im Bereich der Spracherkennung und auch in anderen Verwendungsbereichen eine wichtige Rolle. Hierfür besitzen themenspezifische Sprachmodelle eine sehr gute Leistung. Aber die Trainingsdaten sind nicht für allen Themenbereiche ausreichend, um ein gutes Sprachmodell aufzubauen. Deswegen wird eine Methode für die Vor-Adaption der Trainingsdaten eines Sprachmodells benötigt. Im Folgenden wird vorgestellt, wie die nützlichen Utterances aus den umfangreichen unklassifizierten Dokumenten als die neuen Trainingsdaten extrahiert werden können. Diese Methode berücksichtigt nicht nur die Perplexität einer Utterance sondern auch die Klassifikation des Dokumentes, zu dem diese Utterance gehört, um die Utterances zu bewerten. Mit diesen Bewertungen kann bestimmt werden, ob die Utterances als neue Trainingsdaten extrahiert werden können. In dieser Arbeit werden ein paar Versuche durchgeführt, um die am besten geeigneten Parameter zu bestimmen. Außerdem wird die Evaluation dieser Methode auf Basis von den schon festgelegten Parametern gemacht, um zu sehen, wie viel diese themenspezifische Vor-Adaptierungsmethode ein themenspezifisches Sprachmodell verbessern kann.

Inhaltsverzeichnis

Abstrakt v

1. Einleitung 1
1.1. Motivation . . . 3
1.2. Überblick . . . 3

2. Grundlagen 5
2.1. Sprachmodell und Spracherkennung . . . 5
2.1.1. N-gram . . . 6
2.1.2. Smoothing Technik . . . 6
2.1.3. Bewertungsweise der Qualität eines Sprachmodell . . . 7
2.2. Text Klassifikation . . . 8
2.2.1. Definition von Textklassifikation . . . 8
2.2.2. Maschinelles Lernen(ML) für die Textklassifikation . . . 9
2.2.3. Vorgehensweise zur Textklassifikation . . . 9
2.2.3.1. Vorbereitung des Dokuments . . . 10
2.2.3.2. Feature Selection . . . 10
2.2.3.3. Indizierung des Dokuments . . . 10
2.3. Related Work . . . 11
2.4. Verwendete Software . . . 12
2.4.1. SRILM . . . 12
2.4.2. Rainbow . . . 13

3. Entwurf 17
3.1. Vorbereitung der Daten . . . 19
3.1.1. Bereinigung der Trainingsdaten . . . 20
3.1.1.1. Bereinigung der Dokumente vom Topic Language Model . 20
3.1.1.2. Bereinigung der unklassifizierten Dokumente . . . 20
3.1.2. Weitere Verarbeitung der Daten . . . 21
3.2. Trainieren durch die unklassifizierten Dokumente . . . 22
3.3. Neues Aufbauen des Sprachmodells und die entsprechende Bewertung . . . 23

4. Versuchsaufbau 27
4.1. Experimente mit neuem Aufbauen des Sprachmodells . . . 28
4.2. Versuche mit den varianten Threshhold . . . 29
4.3. Experimente mit neuem Aufbauen des Klassifikationsmodells . . . 29
4.4. Evaluierungsversuche mit den schon bestimmten Parameter . . . 30

5. Ergebnisse 33
5.1. Evaluation der Versuche mit neuem Aufbauen der Sprachmodelle . . . 33
5.2. Ergebnisse der Versuche nach dem Parameter Threshold . . . 35
5.3. Experimente mit neuem Aufbauen des Klassifikationsmodells . . . 38

5.4. Evaluation der Methoden mit den schon bestimmten Parameter 39

6. Zusammenfassung **43**

Literaturverzeichnis **45**

Anhang **47**

A. Tabellen der Versuchsergebnisse 47

1. Einleitung

Mit der rapiden Entwicklung von Technik strebt man immer nach bequemerem Leben mit hoher Qualität. Wir möchten immer die anstrengenden Arbeiten und die Hilfstätigkeiten den Maschinen überlassen. Unter dieser Erwartung wurden Roboter erfunden, die nicht nur Industrieroboter, Serviceroboter, sondern auch humanoide Roboter umfassen, die in unterschiedlichen Bereichen der Menschheit dienen. Man war nicht mehr zufrieden mit der Manipulation der Maschinen durch den Computer mit Maus und Tastatur. Wir möchten mit dem Roboter kommunizieren, ohne Eingaben durch Tastatur. Der Computer soll uns "**hören**", gut "**verstehen**" und soll sich gemäß dem Inhalt unserer Befehle "**verhalten**".

Damit der Computer uns hören und verstehen kann, muss das Konzept **Spracherkennung** eingeführt werden. Ein Spracherkennungssystem kann die akustischen Signale zu reinen Texten umwandeln, wie die Abbildung 1.1 zeigt. Danach können die erkannten Texte nach Analysen und Bearbeitung vom Computer verstanden werden. Dabei stellen $o_1 o_2 \dots o_\tau$ die **Feature Vektoren** dar, die aus dem empfangenen akustischen Signal durch Vor-Verarbeitung umgewandelt werden. Und $w_1 w_2 \dots w_n$ bezeichnet die durch Dekodierung geschätzten Wortfolge, die mit größter Wahrscheinlichkeit als die Quelle von der akustischen Signal gesprochen wird.

$$P(w_1 w_2 \dots w_n | o_1 o_2 \dots o_\tau) = \frac{P(o_1 o_2 \dots o_\tau | w_1 w_2 \dots w_n) \cdot P(w_1 w_2 \dots w_n)}{P(o_1 o_2 \dots o_\tau)} \tag{1.1}$$

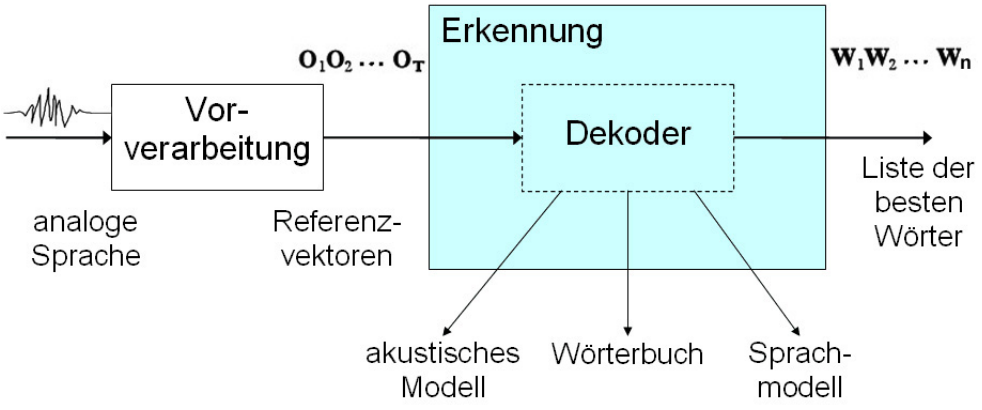


Abbildung 1.1.: Spracherkennungssystem nach Alexander Waibel[wikb]

Formel(1.1) bezeichnet die Formel von **Bayestheorem**, um die Wahrscheinlichkeit einer möglichen Wortfolge $w_1w_2 \dots w_n$ zu berechnen. Hier ist $P(w_1w_2 \dots w_n)$ die A-priori Wahrscheinlichkeit für das Eintreten einer Wortfolge $w_1w_2 \dots w_n$, und $P(o_1o_2 \dots o_\tau|w_1w_2 \dots w_n)$ ist die bedingte Wahrscheinlichkeit für das akustische Signal $o_1o_2 \dots o_\tau$ unter der Bedingung, dass die Wortfolge $w_1w_2 \dots w_n$ eingetretten ist. $P(w_1w_2 \dots w_n|o_1o_2 \dots o_\tau)$ ist die A-posteriori Wahrscheinlichkeit, nämlich wenn das Eintreten eines akustischen Signal $o_1o_2 \dots o_\tau$ schon bekannt ist, wie wahrscheinlich wird dieses Eintreten des Signals durch eine Wortfolge $w_1w_2 \dots w_n$ verursachtet. Dazu sind alle möglichen Wortfolge aus dem identischen empfangenen akustischen Signal, deswegen ist $P(o_1o_2 \dots o_\tau)$ für sie gleich, so kann es weggelassen werden. Dann kann die Formel(1.1) zu der Formel(1.2) vereinfacht werden. Dabei wird $P(o_1o_2 \dots o_\tau|w_1w_2 \dots w_n)$ **akustisches Modell** genannt und $P(w_1w_2 \dots w_n)$ wird **Sprachmodell** genannt.

$$P(w_1w_2 \dots w_n|o_1o_2 \dots o_\tau) = P(o_1o_2 \dots o_\tau|w_1w_2 \dots w_n) \cdot P(w_1w_2 \dots w_n) \tag{1.2}$$

Nach der **Markov Assumption** kann das **Sprachmodell** wie in der Formel(1.3) beschrieben werden. Hier ist das n-te Wort von aller vorher eingetretenen Wörter abhängig, aber es wird zu aufwandig die Wahrscheinlichkeit zu berechnen. Deswegen wird das n-gram Modell gewöhnlich benutzt um ein Sprachmodell aufzubauen. Das bedeutet, dass das n-te Wort nur von den vorherigen n-1 Wörter abhängig ist. Detailierte Erklärung von n-gram und das entsprende Problem werden im Kapitel 2.1.1 wieder diskutiert.

$$P(w_1w_2 \dots w_n) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_1w_2) \dots P(w_n|w_{n-2}w_{n-1}) \tag{1.3}$$

Diese Formeln können sowohl im Bereich der Spracherkennung als auch in anderer Bereiche verwendet werden.Wenn $o_1o_2 \dots o_\tau$ nicht als eine Folge von einem akustischen Signal, sondern als eine Folge von englischen Wörtern angesehen wird und außerdem $w_1w_2 \dots w_n$ nicht als die zu erkennende Wortfolge, sondern als eine übersetzte Wortfolge in einer bestimmten Sprache, dann wird dieses Problem von der Spracherkennung zu maschinellen Übersetzung umgesetzt. $P(o_1o_2 \dots o_\tau|w_1w_2 \dots w_n)$ wird im diesen Kontext für **Übersetzungsmodell** genannt. Und auch wenn wir $o_1o_2 \dots o_\tau$ als eine zu überprüfende Wortfolge ansehen, während $w_1w_2 \dots w_n$ als eine standardisierte Wortfolge angesehen wird, dann ist das Problem schon zu dem Problem **Text Korrektur** umgesetzt geworden, wobei $P(o_1o_2 \dots o_\tau|w_1w_2 \dots w_n)$ für **Korrekturmodell** steht.

Aus diesen Beispielen kann gesehen werden, dass ein **Sprachmodell** in unterschiedlichen Bereichen verwendet werden kann. Es spielt nicht nur eine wichtige Rolle, wenn wir einen Roboter durch einen akustischen Befehl manipulieren möchten(Spracherkennung), wenn ein Student eine Vorlesung, die nicht in seiner Muttersprache gehalten wird, verstehen möchte(Maschinelle Übersetzung) oder eine Ausarbeitung automatisch korrigieren(Textkorrektur) lassen möchte, sogar wenn wir im Internet surfen, um eine Nachricht zu suchen(Information Retrieval). Deswegen brauchen wir immer ein gutes Sprachmodell, um diese Applikationen besser zu unterstützen.

Wegen dieser breiten Anwendungen von Sprachmodell kann es im Bezug auf verschiedene Themenbereiche sein. Wenn wir den Roboter in der Küche uns helfen möchten, die Wörter wir möglich zu sagen sind oft relevant mit solchen Wortfolge z.B. ”Kühlschrank öffnen”, ”Backofen schließen” ,”Teller” und so weiter... Im Gegensatz werden die Wörter ”Kino”,”neuronales netz” oder ”Tischtennis” selten eintreten. Für ein allgemeines Sprachmodell kann es ganz ”tolerant” für die Wörter in unterschiedlicher Bereiche sein, aber es ist leider nicht ”exzellent” für einen spezifischen Bereich. Damit wird ein Sprachmodell normalerweise themenspezifisch aufgebaut werden. In diesem Fall wird das Spracherkennung Problem zuerst in eine Domain verteilt, dann wird es in diesem spezifischen Themenbereich weiter bearbeitet und erkannt, so dass die akustische Signale genauer zu reine Texte umwandelt werden können.

1.1. Motivation

Wie oben genannt, spielt ein gutes Sprachmodell eine wichtige Rolle im Bereich der Spracherkennung und auch in anderen Verwendungsbereichen. Hierfür besitzen themenspezifische Sprachmodelle eine sehr gute Leistung. Aber das Problem ist, dass es nicht für allen Themenbereiche ausreichende Trainingsdateien gibt, um ein gutes Sprachmodell aufzubauen. Unter dieser Bedingung können die umfangreichen Texte von Archive der Zeitungen, der Sendungen order der Bibliotheken im World Wide Web(WWW) berücksichtigt werden. Diese Texte umfassen meistens entsprechende Formatsymbole z.B."<HEADLINE>","<P>",";; LABEL" usw. und werden noch nicht zu einem spezifischen Themenbereich klassifiziert, deswegen können die Texte aus WWW nicht direkt als Trainingsdateien für ein Sprachmodell benutzt werden. Die Formatsymbole dieser Texte müssten zuerst wegglassen werden, um die Analyse der Texte nicht zu stören. Und danach werden die Texte abhängig von der schon extierten Trainingsbeispielen analysiert, um die "nützlichen" Inhalte innerhalb der Texte als neue Trainingsdateien zu extrahieren.

Für die Themenbereiche mit weniger Trainingsdateien muss eine Methode entworfen werden, um zu entscheiden, welche Inhalte aus eines Textes können als themenrelevante Daten zu den Trainingsbeispielen zugefügt werden. Die Motivation meiner Studienarbeit liegt darin, wie die Anzahl an Trainingsdateien von einem themenspezifischen Sprachmodell vergrößert werden kann, sodass dieses Sprachmodell eine bessere Qualität erhalten kann.

1.2. Überblick

In Kapitel 2 wird die für diese Arbeit relevante Theorie vorgestellt. Dazu wird die Theorie von **Sprachmodell und Spracherkennung**(2.1), sowie **Textklassifikation**(2.2) erklärt. Außerdem werden die zwei in dieser Arbeit benutzten Toolkits **SRILM**(2.4.1)und **Rainbow**(2.4.2) detaillierter diskutiert. In Kapitel 3 wird der Entwurf, wie die nützlichen Daten für einen bestimmten Themenbereich aus den Texten unbekannter Kategorien extrahiert werden können, vorgestellt. Das Ausprobieren der Parameter und die in Experimente bestimmten besten Parameter für die Voradaptionsmethode werden in Kapitel 4 diskutiert. In Kapitel 5 werden die Versuchsergebnisse der Parameter und die Evaluation der Sprachmodelle vor und nach der Voradaption angegeben. Diese Arbeit wird in Kapitel 6 zusammengefasst.

2. Grundlagen

2.1. Sprachmodell und Spracherkennung

Ein Sprachmodell versucht, die Wahrscheinlichkeit bestimmter Wortfolge zu bestimmen und dadurch falsche oder unwahrscheinliche Hypothesen auszuschließen oder die Wahrscheinlichsten Hypothesen auszuwählen. Sprachmodelle werden normalerweise im Bereich der (automatischen) Spracherkennung, Maschinellen Übersetzung oder Information Retrieval verwendet.
In der Spracherkennung wird die wahrscheinlichsten Wortfolge mit der Formel [Stü09]:

$$\hat{\mathrm{W}} = arg\max_{W} P(W|X) = arg\max_{W} \frac{P(X|W) \cdot P(W)}{P(X)} \tag{2.1}$$

gefunden. Dabei bezeichnet P(X|W) die Wahrscheinlichkeit des Eintritts einer akustischen Äußerung X unter der Bedingung, dass die Wortfolge W ausgesprochen wird. P(W) bezeichnet die Wahrscheinlichkeit einer Wortfolge, die in einem Sprachmodell berechnet werden kann, während P(X) die Wahrscheinlichkeit bezeichnet, dass eine akustische Beobachtung X eintritt. Dann bezeichnet das akustischen Model P(X|W), wie wahrscheinlich eine bestimmte akustische Beobachtung X unter der Bedingung einer Wortfolge W sein soll. Dazu bezeichnet $\hat{\mathrm{W}}$=argmax$_W$ $P(W|X)$ dann diejenige Wortfolge W, die unter der

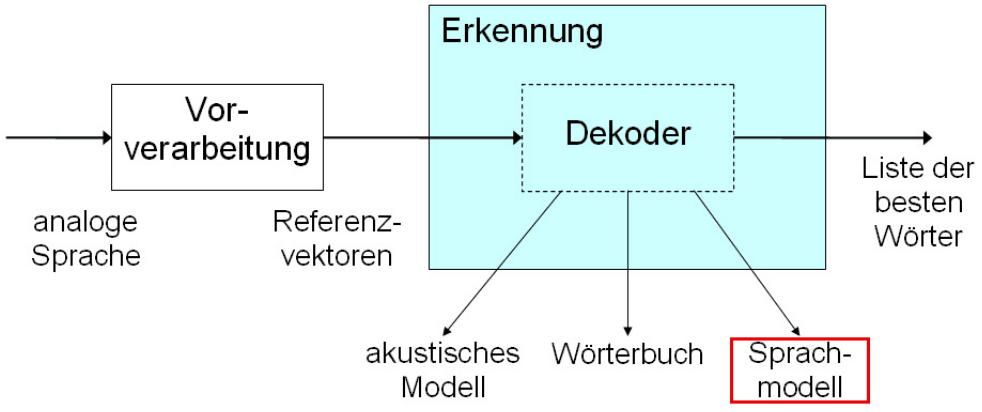


Abbildung 2.1.: Spracherkennung System nach Alexander Waibel[wikb]

gegebenen Beobachtung X am wahrscheinlichsten ist und somit der wahrscheinlichsten Hypothese für die tatsächliche Wortfolge entspricht. P(X) ist konstant für alle W, wodurch sich (2.1) vereinfacht zu [Stü09]:

$$\hat{\mathrm{W}} = arg\max_{W} P(X|W) \cdot P(W) \tag{2.2}$$

Damit kann man sehen, dass ein Sprachmodell eine ganz wichtige Rolle spielt. Um ein Sprachmodell zu erstellen kann entweder ein Grammatikmodell oder ein statistisches Modell eingesetzt werden.

Ein Grammatikmodell hat den Vorteil, dass keine bzw. nur sehr wenige Trainingsdaten vorhanden sein müssen. Aber gleichzeitig hat es den Nachteil, dass die möglichen Äußerungen beim Entwurf der Grammatik bekannt sein müssen, sonst können sie nicht erfolgreich erkannt werden.

Für ein statistisches Sprachmodell ist kein Vorwissen darüber, wie die Wörter zu kombinieren sind, nötig. Aber in diesem Fall werden die umfangreiche Trainingsdaten benötigt, um die Wahrscheinlichkeit einer bestimmten Wortfolge zu bestimmen. Dazu wird gewöhnlich ein N-gram Sprachmodell benutzt, was im Folgenden weiter erklärt wird.

2.1.1. N-gram

Ein Sprachmodell berechnet die Wahrscheinlichkeit einer Wortfolge W=$w_1, w_2 \dots w_i$ mit der Formel [Stü09]:

$$P(w_1 \dots w_i) = P(w_1) \times P(w_2|w_1) \times \dots \times P(w_i|w_1 \dots w_{i-1}) \tag{2.3}$$

Hier sind $w_1 \dots w_{i-1}$ die Kontexte des Wortes w_i. Angenommen, dass das Vokabular aus 10.000 Wörtern besteht und die durchschnittliche Satzlänge 25 Wörter ist, dann kann die Anzahl der möglichen Sätze $10.000^{25} = 10^{100}$ erreichen. Wegen dieses großen Aufwands ist es unmöglich P(w|Kontexte) für jeden Kontext zu schätzen. Unter dieser Bedingung wird in der Praxis nur eine beschränkte Länge von Kontexten berücksichtigt.

Ein N-gram Sprachmodell schätzt die Wahrscheinlichkeit einer Wortfolge unter Berücksichtigung der vorherigen N−1 Wörter. Die Wahrscheinlichkeit für ein Wort w_i mit dem Kontext H $w_1 \dots w_{i-1}$ wird geschätzt durch [wika]:

$$p(w_i|H) \approx P(w_1) \times P(w_2|w_1) \times P(w_3|w_1w_2) \times \dots \times P(w_i|w_{i-(N-1)} \dots w_{i-1}) \tag{2.4}$$

Das Unigram(N=1) ist das einfachste N-gram Sprachmodell, aber es kann die Reihnfolge des Eintritts von Wörtern nicht widerspiegeln. Im Gegensatz dazu werden Bigrame(N=2) und Trigrame(N=3) sehr oft verwendet, weil sie einerseits die Kombinationen der Wörter widerspiegeln können und andererseits der Rechenaufwand nicht so groß ist. Nimmt man ein Trigram als Beispiel, bedeutet es in einem Trigram Sprachmodell, dass nur die letzten zwei Wörter vor dem aktuellen Wort berücksichtigt werden [wika].

$$P(w_1 \dots w_i) \approx P(w_1) \times P(w_2|w_1) \times P(w_3|w_1w_2) \times \dots \times P(w_i|w_{i-2}w_{i-1}) \tag{2.5}$$

2.1.2. Smoothing Technik

Die Häufigkeit des Auftretens einer Wortfolge $w_1 \dots w_i$ in den Trainingsdaten wird mit # bezeichnet. Dann lässt sich die Wahrscheinlichkeit aus den Trainingsdaten wie folgt schätzen [wika]:

$$P(w_i|w_{i-(N-1)} \dots w_{i-1}) = \frac{\#(w_{i-(N-1)}, \dots w_{i-1}, w_i)}{\#(w_{i-(N-1)}, \dots w_{i-1})} \tag{2.6}$$

Angenommen, dass trotzdem in einer große Trainingsdatenmenge eine bestimmte Wortfolge $P(w_i|Histroy)$ z.B. P(Ying|read a book) aber nie aufgetreten ist, dann kommt es zu einem Problem, dass obwohl P(History) z.B P(read a book) ganz hoch ist, die Wahrscheinlichkeit der ganzen Wortfolge P(read a book Ying) auch 0 wird. Egal wie groß die Trainingsdaten ist, kann es nicht gewährleistet werden, dass eine beliebige Wortfolge darin bestimmt auftreten wird.

Diese Bedingung ist ganz schlecht, und kann in folgendem Beispiel gesehen werden. Abhängig von der Formel $\hat{W}$=arg $\max_W P(X|W) \cdot P(W)$, die die Formel(2.1) vereinfacht, kann gesehen werden, wie wahrscheinlich eine Hypothese S: *read a book Ying* unter einem akustischen Signal *A* ist. Wenn in einem Sprachmodell P(S) nämlich P(read a book Ying) = 0 beträgt, dann wird P(S|A) = $P(A|S) \cdot P(S)$ auf jeden Fall 0 betragen, egal wie hoch P(A|S) ist. Das bedeutet, dass diese Wortfolge: *read a book Ying* unter dieser Bedingung nie erkannt werden kann und damit niemals als eine Hypothese berücksichtigt werden kann. Deswegen wird im Verfahren der Spracherkennung ein Fehler eintreten, wenn P(S) = 0 ist.

Um dieses Problem zu lösen wird das Konzept "Smoothing" eingeführt. Es zielt auf ein robusteres Sprachmodell ab. Die Hauptaufgabe des Smoothing ist die Wahrscheinlichkeiten der Wortfolgen neu zu verteilen, um die Bedingung P(S) = 0 zu vermeiden.

In der Literatur [Che98] sind einige Smoothing Methoden beschrieben und analysiert. Die einfachste Smoothing Methode ist "additive Smoothing" [Lid20, Joh32, Jef48]. Sie nimmt an, dass jede n-gram Wortfolge δ mal mehr aufgetreten ist, wobei normalerweise $0 < \delta \leq 1$. Hier zeigt die Formel 2.7, wie additive Smoothing funktioniert.

$$P_{add}(\omega_i|\omega_{i-n+1}^{i-1}) = \frac{\delta + c(\omega_{i-n+1}^{i})}{\delta|V| + \sum_{\omega_i} c(\omega_{i-n+1}^{i})} \tag{2.7}$$

Dabei bezeichnet c, wievielmal die entsprechende Wortfolgen aufgetreten sind, und |V| bezeichnet die Anzahl der Wörter im Vokabular V, welche für das Aufbauen des Sprachmodells benutzt wird.

Außer der obigen Technik gibt es noch die lineare Interpolation(lineare Glättung), welche z.B. die Trigramm-, Bigramm- und Unigramm-Wahrscheinlichkeiten zu eine einheitliche Form interpoliert. Die Formel [Stü09] zeigt, wie die verallgemeinernde N-Gramm Wahrscheinlichkeit interpoliert wird, dazu bezeichnet λ das Gewicht des jeweiligen, zu interpolierenden Items und O die Anzahl dieser Koeffizienten λ_j.

$$P'(w_i|w_{i-n+1}^{i}) = \sum_{j=2}^{O} \lambda_j P(w_i|w_{i-j+1}^{i}) + \lambda_1 P(w_i), \quad \sum_{j=1}^{O} \lambda_j = 1 \tag{2.8}$$

Es gibt noch viele Smoothing Methoden, wie Good-Turing, Katz Smoothing, Kneser-Ney Smoothing usw. , die in der Literatur [Che98] vorgestellt wurden.

2.1.3. Bewertungsweise der Qualität eines Sprachmodell

Um die Qualität eines Spracherkennungssystem zu bewerten bzw. die Erkennung mit zwei Sprachmodelle LM1 und LM2 zu vergleichen kann normalerweise die *WER(Word Error Rate)* benutzt werden [wikc].

$$WER = \frac{S + D + I}{N} \tag{2.9}$$

- S: Anzahl von Substitutionen
- D: Anzahl von Deletionen

- I: Anzahl von Insertionen
- N: Anzahl der Wörter in einer Referenz

Es ist anschaulich, dass je kleiner die entsprechende WER ist, desto besser ein Sprachmodell ist. Zwar ist es möglich mit diesem Maß die Qualität unterschiedlicher Sprachmodelle zu vergleichen, aber es ist teuer, zeitintensiv und sogar schwierig die Qualität eines Sprachmodell direkt quantitativ widerzuspiegeln.

Dazu soll ein Maß eingefügt werden, das ist die sogenannte *Perplexität(PPL)*. Für dieses Maß muss unvermeidbar zuerst das Konzept *"Entropie"* vorgestellt werden, die die Basis der Informationstheorie ist, ein Maß für den mittleren Informationsgehalt eines Zeichens. Formel(2.10) zeigt die Definition der *Entropie*. Sei c_i ein Wort innerhalb einer Wortfolge $X = c_1, c_2, \ldots c_m$ und p_i die Wahrscheinlichkeit des jeweiligen Wortes c_i. Damit ist der Informationsgehalt von c_i : $\mathrm{I} = -\log_2 p_i$ und H(X) als der Erwartungswert des Informationsgehalts.

$$H(X) = \mathbb{E}[-\log_2(p(X))] = -\sum_{i=1}^{m} p_i \cdot \log_2 p_i \tag{2.10}$$

Entropie spiegelt die Unbestimmtheit eines Zeichens(Ereignis) wieder. Je höher die Entropie ist, desto mehr Information wird gebraucht, um ein Zeichen(Ereignis) zu bestimmen.

Die Perplexität ist definiert als :

$$PPL = 2^{H(X)} \tag{2.11}$$

Hier wird zwei als Basis verwendet. Perplexität stellt die Anzahl der Verzweigungen entsprechend eines Ereignis dar. Perplexität bedeutet, aus wie vielen Möglichkeiten kann ein Wort vorhersagen, wenn eine Historie von diesem Wort gegeben wurde. Dadurch kann die Qualität eines Sprachmodell bewertet werden. Ein Sprachmodell ist besser, wenn dessen Perplexität (unter Überprüfungen mit gleichen Testdateien) niedriger ist. Die Testdateien sind im Prinzip repräsentativ für die Texte von einem Themenbereich, in dem das entsprechende Sprachmodell aufgebaut wird. Ein Sprachmodell mit weniger Perplexität bedeutet, dass es für dieses Sprachmodell weniger Verzweigungen benötigt werden, um ein Wortfolge oder ein Dokument zu bestimmen. Perplexität hat gegenüber WER die Vorteile, dass die Optimierung der Perplexität unabhängig von einem Spracherkenner gemacht werden kann und die Perplexität häufig auch einer einfachen, direkten Optimierung zugänglich ist. Die Ergebnisse von WER sind unterschiedlich für unterschiedliche Aufgaben und abhängig von dem Spracherkenner. Deswegen wird in dieser Arbeit Perplexität als das Maß, um zu überprüfen, ob ein Sprachmodell nach der Bearbeitung durch die entworfenen Methode schon verbessert wird, eingesetzt. Für das themenspezifische Sprachmodell bezieht es sich immer auf die Klassifizierung der Texte in einer spezifischen Domain, nämlich das Problem der Textklassifikation.

2.2. Text Klassifikation

2.2.1. Definition von Textklassifikation

Seit den Achtziger Jahren des letzten Jahrhunderts hat die Informationstechnologie wegen einer rapiden Entwicklung des Internets unser Leben ganz viel beeinflußt. Heute muss man nicht dafür sorgen, die Informationen zu bekommen, stattdessen muss man dafür sorgen, die nützlichen Informationen zu extrahieren.

Die automatische Textklassifikation kann uns dabei helfen, die umfangreichen digitalen Dokumenten zu einigen vordefinierten Klassen zu verteilen, sodass die dazu gehörigen

Informationen wegen des spezifischen Vorwissens schneller und mit weniger Aufwand bearbeitet werden können. Unter dieser Bedingung hat die Wichtigkeit der Textklassifikation sich immer abgezeichnet.

Textklassifikation ist die Aufgabe einem Paar $\langle d_j, c_i \rangle \in D \times C$ ein Boolean Wert zuzuordnen. Hier steht D für die Menge der Dokumente und C für die Menge einiger vordefinierter Kategorien. Wenn ein Dokument d_j zu einer Kategorie c_i gehört, wird der Wert T(True) zugeordnet. Der Wert F(False) wird zugeordnet, wenn ein Dokument d_j nicht zur Kategorie c_i gehört. Genauer gesagt, Textklassifikation ist ein Verfahren um der unbekannten Funktion $\Phi : D \times C \rightarrow \{T, F\}$ (beschreibt wie die Dokumente tatsächlich klassifiziert werden sollen) mit einem sogenannten Klassifier $\Phi' : D \times C \rightarrow \{T, F\}$ anzunähern.

2.2.2. Maschinelles Lernen(ML) für die Textklassifikation

Zusammenfassend gesagt, gibt es zwei Arten um die Textklassifikation durchzuführen. Eine Art ist Knowledge Engineering (KE), die in den Achziger Jahren am populäristen (zumindest im Operationsbereich) war, und Machine Learning (ML), das seit den Neunziger Jahren eine immer wichtigere Rolle im Bereich der Textklassifikation gespielt hat.

Für KE sind eine Menge von manuellen von Experten definierten Regeln nötig. Dadurch wird entschieden, welche Dokumente zu welcher Kategorie gehören. In diesem Fall sind Beiträge der Wissenschaftler und der Experten in der jeweiligen Domain benötigt. Deswegen ist die Aufgabe der Textklassifikation mit KE ganz aufwändig zu bearbeiten und auch sehr schwierig zu aktualisieren, wenn neue Domains hinzugefügt werden.

Für Maschinelles Lernen wird eine Menge von bereits manuell klassifizierten Dokumenten benötigt. Wenn die Vorkenntnisse der Kategorien und die entsprechenden trainierenden Beispiele vorgegeben sind, ist es ein induktiver Prozess den Klassifier automatisch zu bilden.

Der Unterschied zwischen ML und KE und auch die Vorteile von ML bestehen darin, dass ML nicht darauf fokussiert ist, wie ein Klassifier zu konstruieren sondern wie ein Klassifier automatisch zu bilden ist. Das Bilden eines Klassifiers mit Maschinellem Lernen liefert eine sehr gute Leistung bei der automatischen Klassifikation.

Um Maschinelles Lernen zu benutzen, muss ein Korpus $\Omega = \{d_1, ..., d_{|\Omega|}\}$ von Dokumenten vorhanden sein, der schon nach der Menge der Klassen $C = \{c_1, ..., c_{|C|}\}$ vorklassifiziert worden ist. Das bedeutet, dass jeder Wert von der Funktion $\Phi : D \times C \rightarrow \{T, F\}$ für das jeweiliges Paar $\langle d_j, c_i \rangle \in \Omega \times C$ schon bekannt ist. Ein Dokument d_j ist ein positives Beispiel von c_i, wenn $\Phi(d_j, c_i) = T$. Ein Dokument d_j ist ein negatives Beispiel, wenn $\Phi(d_j, c_i) = F$. Für eine Methode des Maschinellen Lernen können drei Sets aus dem Korpus Ω getrennt konstruiert werden, nämlich Training Set, Validation Set und Test Sets.

Ein Training Set $Tr = \{d_1, ..., d_{|Tr|}\}$ ist aus dem Korpus Ω und wird benutzt, um den Klassifier induktiv zu konstruieren. Ein Validation Set $Va = \{d_{|Tr|+1}, ..., d_{|TV|}\}$ wird benutzt, um die Parameter vom Klassifier zu üben und zu optimieren. Ein Test Set $Te = \{d_{|TV|+1}, ..., d_{|\Omega|}\}$ wird benutzt, um die Effektivität des Klassifiers zu bewerten.

Wenn der vom Set Tr trainierte Klassifer auf dem gleichen Set getestet wird, werden die Ergebnisse unreal gut wegen dem "Overfitting" Problem und diese Evaluation wird sinnlos. Aus dem gleichen Grund kann der aus Va optimierte Klassifier auch nicht auf dem Test Set testen. Deswegen müssen diese drei Sets voneinander getrennt sein.

2.2.3. Vorgehensweise zur Textklassifikation

Die wichtigsten Schritte zur Textklassifikation umfassen prinzipiell die Vorbereitung des zu bearbeitenden Dokuments, Dokument Indexing, Features Selection, Klassifier Trainierung

und Klassifier Auswertung. Darin werden Vorbereitung des Dokuments, Features Selection und Indizierung des Dokuments im Folgenden näher vorgestellt.

2.2.3.1. Vorbereitung des Dokuments

Die Texte der Dokumente können von dem Computer nicht direkt eingelesen und bearbeitet werden. Damit müssen zuerst alle Texte zu einer Form umwgeandelt werden, die vom Computer eingelesen und vom Klassifier benutzt werden kann.

Für die Vorbereitung des Dokuments sollten die Stop-words wie Artikel, Präpositionen, Konjunktionen und so weiter weggelassen werden, sodass die Features Dimensionen abnehmen können und auch die Möglichkeit die Inhalte misszuverstehen verringert werden kann.

Auf Deutsch oder auf Englisch sind die Wörter oft in vielen Formen wegen Zeitformen, Plural oder der Deklination. Z.B. für "Apfel" gibt es die Pluralform "Äpfel", für "machen" gibt es die Zeitform "gemacht" und für "schöne Blume" gibt es Delination "schöner Blume" und so weiter... Deswegen ist Stemming im Verfahren der Vorbearbeitung auch nötig.

2.2.3.2. Feature Selection

Nach der Säuberung der Texte sollte aus diesen Texte eine Menge der Features gefunden werden, um diese Texte repräsentativ und einheitlich darzustellen. Feature Selection ist ein Prozess zur Auswahl repräsentativer Vektoren von Dokumenten, um eine gleiche Struktur der Dokumente darzustellen, sodass sie vergleichbar sind. Feature Selection ist auch ein Prozess um die Dimensionen dieser repräsentierten Vektoren zu verringern. Obwohl im Prozess der Vorbearbeitung von Dokumenten die Stop-words schon weggelassen sind, ist die Dimension der Features noch ganz groß. Die große Menge der Features wird einerseits einen großen Aufwand zur Klassifikation und andererseits auch wegen ungenauen extrahierten Informationen aus dem Text ein schlechter Ergebnis der Klassifikation verursachen. Deswegen sollte die Dimension möglichst minimiert werden, unter der Bedingung nicht die Qualität der Klassifikation zu schaden. Das Ziel von Feature Selection ist es, um die Features mit wenigen Informationen, die unwichtigen Features und die nicht relevanten Dokumentes zu löschen, sodass die Anzahl der Features abnehmen kann. Feature Selection ist ein kritischer Schritt im automatischen Textklassifikationssystem.

2.2.3.3. Indizierung des Dokuments

Nach der Auswahl der Features sollte die Texte noch zu einer Form umgewandelt werden, die der Computer verstehen kann. Prinzipiell kann ein Text d_j als ein Vektor $\overrightarrow{d_j} = \langle \omega_{1j}, ..., \omega_{|\tau|j} \rangle$ über das Feature-Gewicht repräsentiert werden. Dabei ist τ eine Menge der Features und $0 < \omega_{kj} < 1$ steht dafür, wie wichtig ein Feature t_k zu einem Dokument d_j ist, um es zu klassifizieren.

Es gibt viele Arten um das Feature-Gewicht zu repräsentieren, wie TF (Term Frequency), TFIDF (Term Frequency and Inverse Document Frequency), MI (Mutual Information) und so weiter... Dabei ist TFIDF am häufigsten und auch am einfachsten zu benutzen.

TFIDF wird im Folgenden definiert : $tfidf(t_k, d_j) = \#(t_k, d_j) \cdot log \frac{|Tr|}{\#Tr(t_k)}$. Dabei bezeichnet $\#(t_k, d_j)$, wie viel Male der Feature k im Dokument j erscheint. $\#Tr(t_k)$ bezeichnet die Anzahl der Dokumente im Training Set, in welchen der Feature k erscheint. Diese Funktion zeigt, dass je größer $\#(t_k, d_j)$ ist, d.h. je häufiger ein Feature in einem Dokument erscheint, desto mehr kann dieser Feature das entsprechende Dokument repräsentieren. Und es zeigt auch, dass je kleiner $\#Tr(t_k)$ ist, d.h. je mehr Dokumente den Feature k enthalten, desto schwächer ist die Fähigkeit dieses Features, Dokumente zu unterscheiden.

2.3. Related Work

Die Methode dieser Arbeit kombiniert die von den Sprachmodellen berechneten Perplexitäten einer Utterance und die Klassifikation eines Dokumentes, zu welchem diese Utterance gehört, um diese Utterance zu bewerten. Einige andere Methoden verbessern das themenspezifische Sprachmodell auch durch die Voradaption von Sprachmodellen, z.B. Moore(2010)[ML10] und Sethy(2005)[SGN05].

Die Methode in der Literatur Moore(2010)[ML10] hat angenommen, dass ausreichend Trainingsdaten eines themenspezifischen Sprachmodells vorhanden sind. Dieses relativ gute Sprachmodell wird zur Bewertung der Textdaten anderer Quellen benutzt, um die nützlichen Daten zu extrahieren und weitergehend das themenspezifische Sprachmodell zu optimieren. Der Unterschied zwischen dieser Methode und den vorherigen Arbeiten wie Gao(2002)[GGLL02] liegt daran, dass zur Bewertung der Textdaten nicht die Perplexität als eine Metrik, sondern die Differenz der Kreuzentropien eines Textes verwendet wird. Diese Kreuzentropien werden entsprechend dem themenspezifischen Sprachmodell und einer zufälligen Untermenge der Background Textdaten, aus denen der zu bewertenden Text kommt, berechnet. Solche Textdaten werden als die neuen themenbezogenen Daten extrahiert, wenn deren Differenz der Kreuzentropien weniger als ein bestimmter Threshold ist. Die Experimente von Moore(2010) verwendeten den englischen Teil der englisch-französischen Texte aus Release v5 vom Europarl Korpus([Koe05]) als themenspezifische Daten, welche die Proceedings des europäischen Parlamentes zwischen dem Jahr 1999 und dem Jahr 2009 umfasst. LDC English Gigaword Third Edition(LDC Catalog No.: LDC2007T07) wurde hier als die Background Textdaten benutzt. Die vom Sprachmodell, das vom ganzen Gigaword Korpus aufgebaut wurde, bewertete Perplexität ist 135. Die beste Baseline Methode in den Versuchen ist Klakow's Methode(2000)[Kla00], die eine Perplexität von 110.8 bzw. einen Verbesserungsgrad von 17.93% erreichen kann. Die Methode Moore(2010) hat in den Versuchen eine Perplexität von 101.9 bzw. einen Verbesserungsgrad von 24.52% erreicht.

Die Methode dieser Arbeit basiert auf der Methode in der Literatur [SGN05]. Eine solche Methode [SGN05] orientierte sich auch an der Lösung zum Problem der mangelnden themenbezogenen Daten eines themenspezifischen Sprachmodells. Aber sie fokussierte sich auf die Webdaten, die viele Tags oder nutzlosen Daten beinhalten können. Deswegen hat sie drei Modelle aufgebaut: themenspezifisches Modell, Background Modell und Reject Modell. Solche hinzukommende Webdaten, die mit dem Rejection Modell hoch oder mit dem themenspezifischen oder Background Sprachmodell ganz niedrig bewertet worden sind, werden nicht akzeptiert. Zur Bewertung der Utterances waren nicht nur die relative Entropie einer Utterance sondern auch die Wahrscheinlichkeit eines Dokumentes, das diese Utterance beinhaltet, berücksichtigt. Die Patient-Arzt Dialoge wurden in den Versuchen als themenspezifischen Textdaten benutzt, während hier ein Sprachmodell, das von den Daten aus SWB, WSJ und dem Gutenberg Projekt interpoliert wurde, als das Background Sprachmodell verwendet wurde. Diese Methode[SGN05] hat eine WER von 24% erreicht, die ein Verbesserungsgrad von 14% im Vergleich mit der WER der Baseline Methode(28%) erreicht hat. Dabei wurde das Baseline Sprachmodell von den themenspezifischen Daten und dem CMU Sprachmodell interpoliert.

Obwohl die Methode dieser Arbeit basierend auf der obigen Methode [SGN05] entworfen worden ist, sie sind voneinander unterschiedlich. Die Methode dieser Arbeit wird das Reject Modell nicht aufbauen und außerdem wird sie die neu hinzukommenden Utterances mit deren Perplexitäten und der Dokumentenwahrscheinlichkeit bewerten, um die nützlichen Textdaten zu erhalten.

2.4. Verwendete Software

In dieser Arbeit bezieht sich die Voradaption des themenspezifischen Sprachmodells auf das Sprachmodell und die Textklassifikation, deswegen wird das Werkzeug SRILM benutzt, um die Daten zum Aufbau eines Sprachmodells zu erhalten. Und die Software Rainbow wird benutzt, um die Textklassifikation der Dokumente durchzuführen.

2.4.1. SRILM

SRILM [SRI] ist ein Werkzeug, die Sprachmodelle aufzubauen und zu verwenden. Dessen vollständigen Name bezeichnet als **Stanford Research Institute Language Modeling Toolkit**. Es war während **Johns Hopkins University/CLSP summer workshops** im Jahr 1995 begründet und ist seit Jahr 1995 unter der Entwicklung in **SRI Speech Technology and Research Laboratory**.

SRILM kann in der Bereiche von Spracherkennung, statistisches Tagging, Segmentation und Maschinellen Übersetzung verwendet werden. Es ist verwendbar nicht nur auf der Unix Plattform sonder auch auf der Windows Plattform. Es umfasst eine Sammlung von C++ Bibliotheken, ausführbare Programme und Hilfe Skripts. Diese ermöglichen das Produzieren eines statistischen Sprachmodells und die darauf laufenden Experimente für die Spracherkennung und andere Applikationen.

Die Hauptaufgabe von SRILM ist die Schätzung und Evaluation eines Sprachmodells zu unterstützen. Hier bedeutet **Schätzung** das Aufbauen eines Sprachmodells aus den Trainingsdaten, und **Evaluation** bedeutet die Bewertung der Testdaten von den vorhandenen Sprachmodelle, die konventionell mit **Perplexität(PPL)** beschrieben wird.

Die hier am meistens verwendeten Programme sind ngram-count und ngram, dabei kann das Kommando(2.12) benutzt werden, um ein Standard-Sprachmodell LM(Trigram nämlich -order 3 mit Good-Turing discounting und Katz backoff für *Smoothing*) aus den Trainingsdaten TRAINDATA aufzubauen.

```
ngram-count -text TRAINDATA -lm LM
```
(2.12)

Das Kommando2.13 kann benutz werden, um einen Testkorpus TESTDATA von einem vorhandenen Sprachmodell LM zu evaluieren. Hier bedeutet -debug 2, dass die Bewertung auf der Wort-Schicht durchgeführt wird, während bezeichnet -debug 1 auf der Satz-Schicht und -debug 0 auf dem ganzen Korpus.

```
ngram -lm LM -ppl TESTDATA -debug 2
```
(2.13)

Das Sprachmodell LM wird als Form(Abbildung 2.2) darstellt.

Nachdem ein Sprachmodell aufgebaut worden ist, kann die Perplexität des Testkorpus mit Kommando((2.14)) berechnet werden. Dazu wird Testkorpus TESTDATA mit PPL(log(P(T), das Produkt der Wahrscheinlichkeiten den allen Sätzen im Korpus TESTDATA) evaluiert. Die Ausgabe der Evaluation wird im File LM.ppl gespeichert.

```
ngram -lm LM -ppl TESTDATA -debug 0 > LM.ppl
```
(2.14)

Die Ergebnisse werden wie im Folgenden beschrieben:

Dabei zeigt die erste Linie die elementaren Informationen des getesteten Files: der Name dieses Files und die Anzahl den Sätzen 784, die Anzahl den Wörten 24055, die **out-of-vocabulary** Wörtern. Während zeigt die zweite Linie die Informationen der Evaluation.

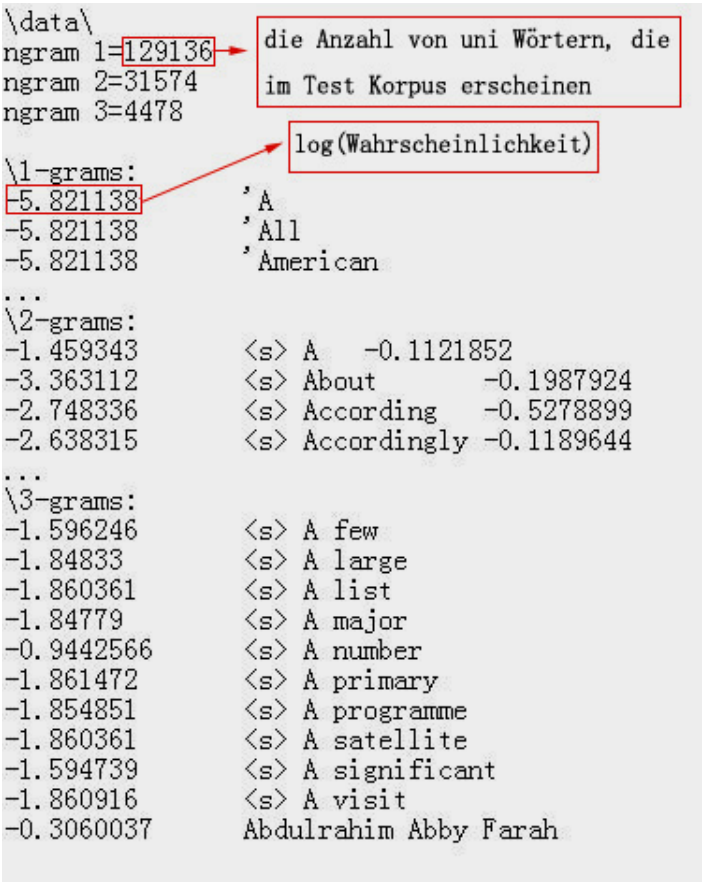


Abbildung 2.2.: Textform eines Sprachmodells

```
file /Test_sets/TLMTest.txt: 784 sentences, 24055 words, 589 OOVs
0 zeroprobs, logprob= -60329.2 ppl= 307.469 ppl1= 372.322
```

Abbildung 2.3.: die von Kommando 2.14 berechneten Ergebnisse

Hier bezeichnet beide ppl und ppl1 die Perplexität, aber die Formeln differenzieren sich wie Folgenden:

$$ppl = 10^{-\frac{\log prob}{Sen+Word}} \qquad ppl1 = 10^{-\frac{\log prob}{Word}} \tag{2.15}$$

Hier bezeichnet Sen die Anzahl der Sätze und Word die Anzahl der Wörter. Trotzdem ist Perplexität im Prinzip als $ppl = 2^{-\sum_{i=1}^{m} p_i \cdot \log_2 p_i}$ definiert, aber für die ppl im SRILM ist der Basis als 10 gesetzt. Der Basis kann von 2 zu 10 geändert werden, weil für $\log prob$ es auch unter dem Basis 10 berechnet wird, nämlich $\log_{10} prob$. Weitere Informationen über SRILM werden in folgenden Kapitel wieder erklärt.

2.4.2. Rainbow

Rainbow[McC] ist als ein Frontend Programm die statistischen Text Klassifikation durchzuführen. Es basiert auf der Bibliothek Bow, die für die Programmierung von statistischen Text Analyse, Sprachmodell und Information Retrieval ganz nützlich ist. Sowohl die Bibliothek Bow als auch die entsprechenden Frontend für Dokument Klassifikation(rainbow), Dokument Retrieval(arrow) und Dokument Klustering(crossbow) sind von **Andrew McCallum** in Carnegie Mellon Universität entwickelt.

In dieser Studienarbeit wird nur Rainbow als ein Werkzeug benutzt, um zu rechnen, wie wahrscheinlich ein Dokument zu einer Klasse gehört .

Rainbow kann die Gewichte der Wörter- Vektoren mit **Naïve Bayes**, **TFIDF** und den anderen Methode anbieten. Für **Smoothing** Problem über die Wahrscheinlichkeiten der Wörter kann die Methode aus **Laplace**, **M-estimates**, **Witten-Bell** und **Good-Turning** ausgewählt werden. Außerdem hat Rainbow noch den Vorteil, dass die Trainingsdaten keine Vorbereitung gebraucht(keine besondere Tags am Anfang oder Ende des Dokumentes nötig) werden, um die Indizierung- Funktion zu laufen. Aber Rainbow kann nicht unter Windows benutzt werden und es wird leider nicht gut dokumentiert.

Das normale Verfahren ein Dokument zu klassifizieren ist zuerst ein Modell mit bestimmter Methode aufzubauen und danach das Dokument gemäß diesem Modell zu klassifizieren.

Um ein Modell aufzubauen, müssen die Trainingsdaten vorher gemäß jeweiliger Klasse unter unterschiedlichen Verzeichnisse gespeichert werden. In dem Kommando (2.16) wird ~/20_ newsgroups/talk.politics.* von Shell zu folgenden Namen der Verzeichnisse umgesetzt: ~/20_ newsgroups/talk.politics.guns, ~/20_ newsgroups/talk.politics.mideast und ~/20_ newsgroups/talk.politics.misc.

$$rainbow \quad -d \quad \tilde{}/model \quad --index \quad \tilde{}/20\_newsgroups/talk.politics.* \tag{2.16}$$

Nach "-d" folgt das Verzeichnis, wo das Modell gespeichert wird, während folgt nach "--index" den Name des Verzeichnis, darin die Trainingsdaten gemäß der unterschiedlichen Klasse in den entsprechenden Subverzeichnissen gespeichert werden. Wenn ohne der Option "-d", wird das aufgebaute Modell unter ~/.rainbow als Standard gespeichert.

Nachdem ein Modell entsprechend den Unterschiedlichen Klassen aufgebaut worden ist, kann ein Dokument mit dem folgenden Kommando klassifiziert werden. Wobei gibt es zwei Hinweise über -- query:
1. Es muss zwischen -- query= und /AbsolutPath/Testdocument keine Leerzeichens geben.
2. Relatives Verzeichnis z.B. ~/Testdocument kann hier nicht benutzt werden.

$$rainbow \quad -d \quad \tilde{}/model \quad --query = /AbsolutPath/Testdocument \tag{2.17}$$

Das Kommando und die entsprechenden Ergebnisse nach dem Aufbauen des Modell werden in der Abbildung(2.4) gezeigt.

```
i13pc236 /home/yxu> ./bow-20020213/rainbow -d ~/Training_Model_tfidf --index ~/T
raining_sets2/* --method=tfidf
Created directory `/home/yxu/Training_Model_tfidf'.
Class `BLM_sets'
  Gathering stats... files : unique-words ::     64 :     4914
Class `TLM_sets'
  Gathering stats... files : unique-words ::     64 :    10570
Setting weights over words:         9
Normalizing weights:         0
```

Abbildung 2.4.: Screenshot vom Aufbauen des Klassifikationsmodells

In der Abbildung(2.5) wird das Kommando gezeigt, mit ihm ein Dokument von einem schon aufgebauten Modell klassifiziert wird. Und die entsprechenden Ergebnisse werden auch darin ausgegeben. Für die Option "-- method=" kann der Wert von nicht nur tfidf sondern auch naivebayes, knn, prind und svm gesetzt werden. Aber die Ergebnisse unterscheiden sich mit einander. Wenn die Option "-- method" nicht konfiguriert wird, dann wird die Methode naivebayes als Standard benutzt.

Es gibt noch viele Optionen, die die Funktionen: Tokenizing,Smoothing, Trennung der Trainingsdaten und Testdaten, und auch die Evaluation eines Batch von Dokumenten,

```
i13pc236 /home/yxu> ./bow-20020213/rainbow -d ~/Training_Model_tfidf --query=test
Loading data files...
Placed remaining 128 documents in the train set:
Setting weights over words:         9
Normalizing weights:         0

BLM_sets 0.03239075467
TLM_sets 0.02018326707
```

Abbildung 2.5.: Klassifikation eines Dokumentes

anbieten. Weitere Informationen und Parameter Konfigurationen können auf der Website Rainbow erhalten werden.

3. Entwurf

Um die Qualität eines Sprachmodells zu garantieren, müssen ausreichende Trainingsdaten vorhanden sein. Aber für manche themenspezifische Sprachmodelle gibt es nur wenige Trainingsdaten. Deswegen ist es nötig aufgrund der umfangreichen unklassifizierten Dokumenten die themenbezogenen Inhalte zu extrahieren. Abbildung 3.1 zeigt den Überblick des ganzen Verlaufs dieser Arbeit. Hier werden zwei Konzepte "Background Language Model"

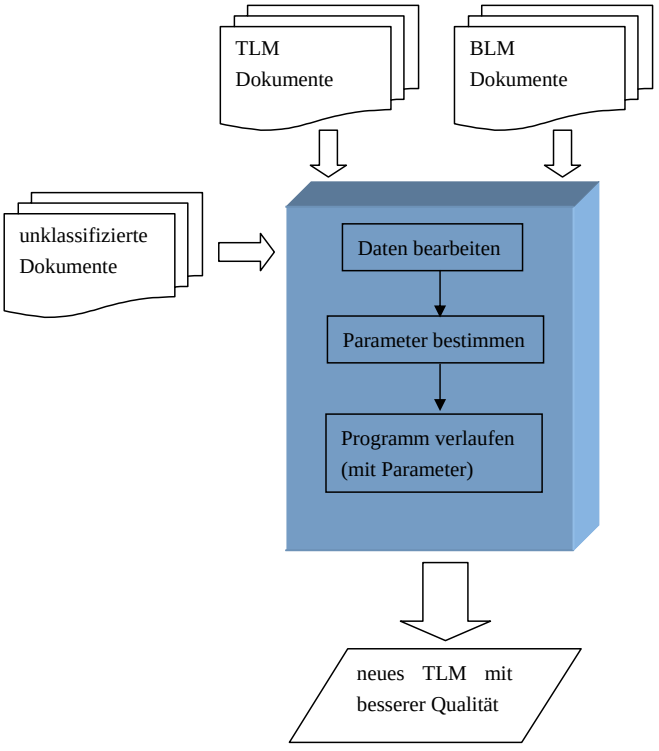
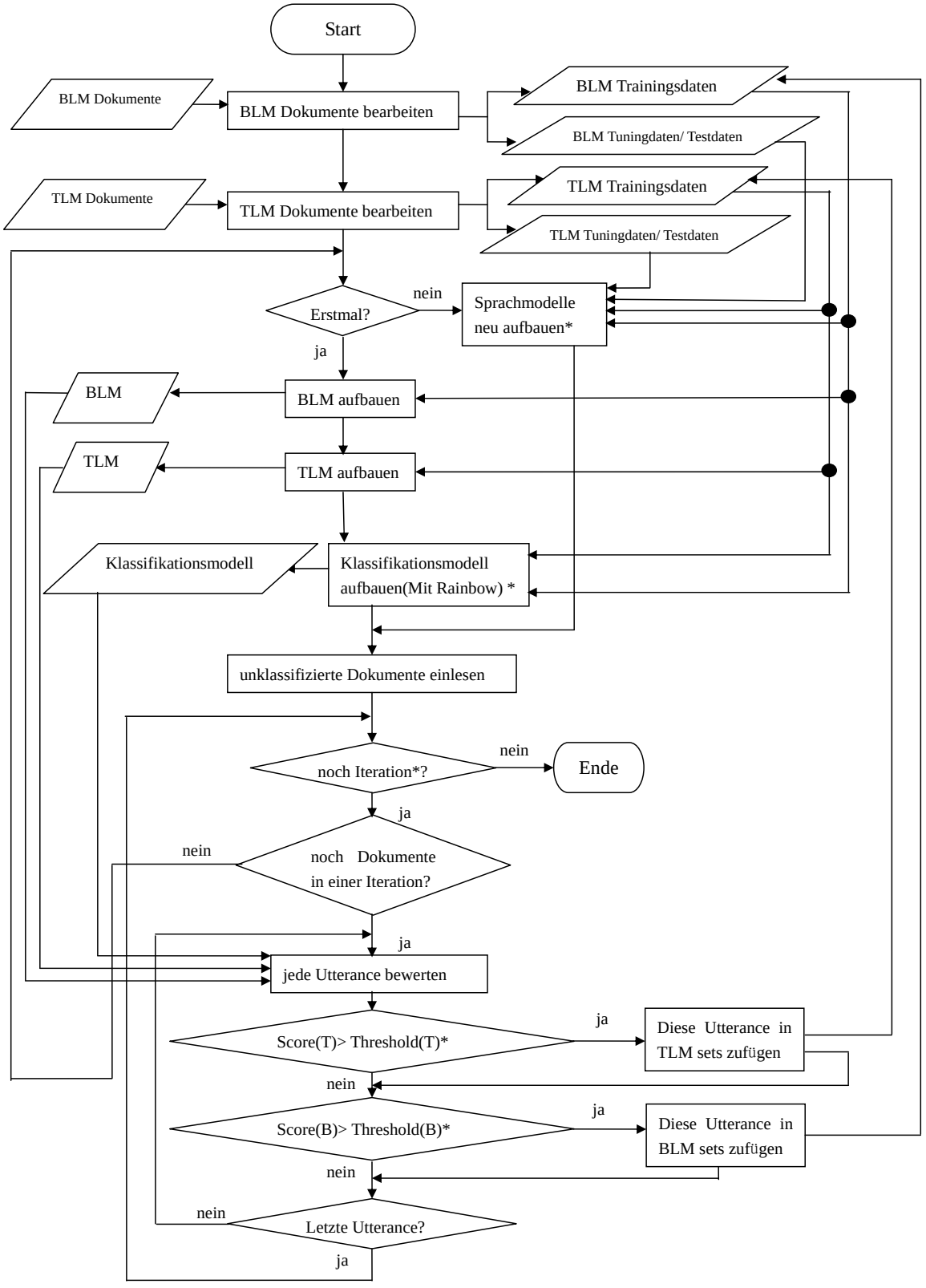


Abbildung 3.1.: Übersicht des Programms

und "Topic Language Model" eingeführt. Ein Background Language Model ist unabhängig von einem bestimmten Thema. Es umfasst die Informationen aus unterschiedlichen Themenbereichen und kann die Struktur der Wörterfolge wiederspiegeln. Das Topic Language Model umfasst die Informationen, die relevant für einen bestimmten Themenbereich sind. Hier werden die Dokumente von Background Language Model(BLM) und die Dokumente



Tuningdaten— benutzt um die Parameter zu bestimmen. Testdaten— verwendet um die Methode dieser Arbeit zu evaluieren. Schritte mit * — die Parameter darin probiert und festgelegt.

Abbildung 3.2.: Ablaufdiagramm des Programms

von Topic Language Model(TLM) als Eingabedaten benutzt, die nach der Bearbeitung in Trainingsdaten, Tuningdaten und Testdaten getrennt werden. Nach der Bearbeitung von BLM und TLM Dokumente werden die unklassifizierten Dokumente auch als Eingaben benutzt. In diesem Prozess werden die Dokumente auf der Utterance-Ebene beobachtet. Das bedeutet, dass jede Utterance eines Dokumentes getestet wird, ob diese Utterance eine gegebene Bedingung gut genug erfüllen kann. Der Vorteil der Methode dieser Arbeit im Vergleich mit den existierenden Arbeiten liegt darin, dass zur Bewertung einer Utterance nicht nur ihre Perplexität sondern auch die Klassifikation des Dokumentes berücksichtigt werden, zu dem die Utterance gehört. Die guten Utterances werden in die Trainingsdateien hinzugefügt. Durch die Vermehrung der Trainingsdateien des TLMs wird ein themenspezifisches Sprachmodell mit der verbesserten Qualität als Ausgabe meiner Arbeit geliefert. Expandierte Prozesse werden in der Abbildung 3.2 weiter beschrieben. Hier werden Tuningdaten in den Versuchen, die die Parameter bestimmen, benutzt. Testdaten werden im Evaluierungsversuch verwendet. Die Prozesse in dem Ablaufdiagramm mit * bedeuten, dass die Parameter in diesen Prozessen probiert und bestimmt werden. Hier bezogen sich die Parameter auf die Schwellwerte von BLM und TLM und auf die Anzahl der Iterationen, welche für das neu aufgebaute Sprachmodell oder Klassifikationsmodell für die Bewertung der Utterances verwendet wird. Der Teil des Prozesses, welcher die Sprachmodelle neu aufbaut, ist sehr kompliziert, so dass er im Unterkapitel 3.3 detailliert demonstriert wird. Die detaillierteren Versuche, um die Parameter zu bestimmen, und die Evaluation unserer Methode werden im Kapitel 4 weiter erklärt.

3.1. Vorbereitung der Daten

Die Eingabedaten können im Verlauf der Versuche leider nicht direkt verwendet werden. Einerseits gibt es viele redundante Inhalte wie Tags, Satzzeichen usw. darin, die weggelassen werden müssen. Diese Inhalte werden die Versuchsergebnisse beeinflussen, wenn sie in den Texten bleiben. Andererseits sollten die Daten in Trainingsdaten, Tuningdaten und Testdaten getrennt werden, um das "Overfitting" Problem zu vermeiden. Deswegen werden

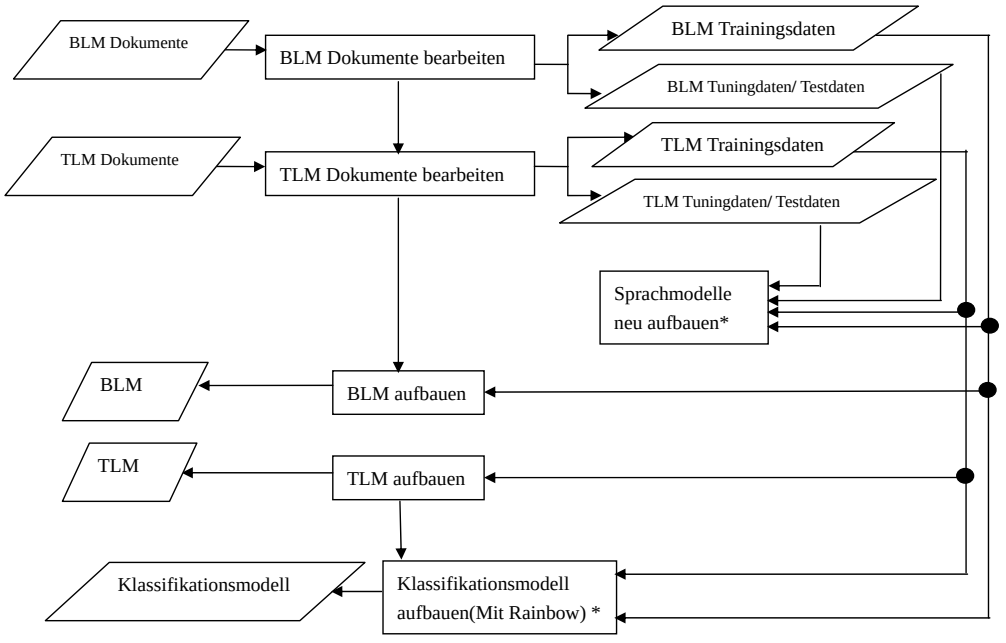


Abbildung 3.3.: Vorbereitung der Daten

die Daten zuerst gereinigt und dann in die drei Untermengen getrennt. Die Abbildung 3.3 ist Teil von dem Ablaufdiagramm 3.2 und zeigt die Hauptaufgaben der Vorbereitung der Dateien und die existierten Beziehungen mit den anderen Teilen des Ablaufdiagrammes. Durch "**BLM(TLM) Dokumente bearbeiten**" werden die BLM(TLM) Dokumente zuerst gereinigt und dann in Trainingsdaten, Tuningdaten und Testdaten getrennt. Trainingsdaten und Tuningdaten werden in den Versuchen, in welchen die Parameter bestimmt werden, benutzt. Trainingsdaten und Testdaten werden in der Evaluation der Methode weiter verwendet, dabei werden auch die schon festgelegten Parameter benutzt.

3.1.1. Bereinigung der Trainingsdaten

Die BLM Trainingsdaten sind hier schon saubere Dokumente, die ohne zusätzliche Kennzeichen oder Satzzeichen sind und in jeder Zeile nur eine Utterance haben. Aber die TLM Dokumente und die unklassifizierten Dokumente sind nur Rohdaten, deren Bereinigungen noch benötigt werden.

3.1.1.1. Bereinigung der Dokumente vom Topic Language Model

Die TLM Trainingsdaten sind Rohdaten, die aus einer Radio-Sendung gesammelt wurden(siehe Abbildung 3.4). Hierfür müssen die Dateien abhängig von den entsprechenden

```
;; Transcriber export by stm.tcl,v 1.19 on mer ao373 26 13:28:15 CEST 2009 with encoding ISO-8859-1
;; transcribed by Megan Richards, version 5 of 090819
;;
;; CATEGORY "0" "" ""
;; LABEL "O" "Overall" "Overall"
;;
;; CATEGORY "1" "Hub4 Focus Conditions" ""
;; LABEL "F0" "Baseline//Broadcast//Speech" ""
;; LABEL "F1" "Spontaneous//Broadcast//Speech" ""
;; LABEL "F2" "Speech Over//Telephone//Channels" ""
;; LABEL "F3" "Speech in the//Presence of//Background Music" ""
;; LABEL "F4" "Speech Under//Degraded//Acoustic Conditions" ""
;; LABEL "F5" "Speech from//Non-Native//Speakers" ""
;; LABEL "FX" "All other speech" ""
;; CATEGORY "2" "Speaker Sex" ""
;; LABEL "female" "Female" ""
;; LABEL "male"   "Male" ""
;; LABEL "unknown"   "Unknown" ""
17438 1 17438_speaker_1 0.000 17.332 <o,f0,female> welcome to the fourth public lecture podcast from the Uni
versity of Bath. [r] Member of Parliament Clare Short, former Cabinet Minister for International Development
, talks about the UK's potential role in helping unify a divided world. organized by the University of Bath
and the Royal Society for the Arts.
17438 1 17438_speaker_2 17.332 30.328 <o,f0,female> [r] the intention of this lecture series is to give prom
inent speakers %% an opportunity to talk on important national and international issues.
```

Abbildung 3.4.: TLM Trainingsdatei

Datenstrukturen bearbeitet werden. In dem Text des Themas sollten die nutzlosen Zeichens wie ";; LABEL...", Zeichen vor und innerhalb "<>" und die Zeichen "%% , ? ! . # ..." gelöscht werden, so dass die Utterances darin vom Werkzeug SRILM bearbeitet werden können und richtige Ergebnisse erhalten werden können. Alle Dokumente hier müssen als separate Texte extrahiert werden und jeder Paragraph darin muss auch als eine separate Utterance herausgelesen werden, um die Score jeder Utterance zu berechnen.

3.1.1.2. Bereinigung der unklassifizierten Dokumente

Die unklassifizierten Dokumente sind aus den Zeitungstexten, in der auch zusätzliche Tags vorliegen, wie in der Abbildung 3.5 gezeigt. Analog zu den Trainingsdaten müssen die Tags hier auch weggelassen werden, um die Dokumente zu säubern. Hierfür werden die Zeichen wie "<DOC id=...>", "<HEADLINE>", "</HEADLINE>", "<TEXT>" "<P>" usw. weggelassen. Die Textinhalte zwischen diesen Zeichen müssen extrahiert werden und

```
<DOC id="XIN_ENG_19950101.0001" type="story" >
<HEADLINE>
Russian Troops Seize Chechen Capital
</HEADLINE>
<DATELINE>
MOSCOW, January 1 (Xinhua)
</DATELINE>
<TEXT>
<P>
Russian troops today seized the Chechen capital Grozny and have fully
kept the city under their control, the Press Service of the Russian
Government announced this night.
</P>
```

Abbildung 3.5.: Datenstruktur der unklassifizierten Dokumente

die Inhalte in jedem Paragraph, die von den Zeichen "<P>" und "</P>" begrenzt werden, werden als eine Utterance gesehen. Die zusätzlichen Zeichen und Satzzeichen von dieser "Utterance" müssen auch weggelassen werden, um nachher ihre Scores zu berechnen. Der Vorbereitungsprozess der unklassifizierten Dokumente ist zwar nicht in der Abbildung 3.3 gezeigt worden, aber er wird während des Einlesens der unklassifizierten Dokumente durchgeführt.

3.1.2. Weitere Verarbeitung der Daten

Nach der Bereinigung der Daten werden die BLM und TLM Dokumente weiter entsprechend in Trainingsdaten, Tuningdaten und Testdaten getrennt. Dabei werden die Trainingsdaten und Tuningdaten in den Versuchen benutzt, um die Parameter zu bestimmen. Die Trainingsdaten und die Testdaten werden im Evaluationsprozess der Methode verwendet. Hier werden die "**Trainingsdaten**" benutzt, um die Sprachmodelle von BLM und TLM aufzubauen. Die "**Tuningdaten/ Testdateien**" werden benutzt, um die Qualität des entsprechenden Sprachmodells zu bewerten, da sie sehr repräsentativ für das zu bewertende Sprachmodell sind. Hier wird zufällig 20% vom Text als die Tuningsdaten benutzt, 20% vom Text wird als die Testdaten extrahiert und die anderen 60% werden als Trainingsdaten verwendet. Im Verlauf des ganzen Programms werden die Trainingsdaten vermehrt, aber die Tuningdaten/ Testdaten sollten dieselben bleiben, um die Qualität der Sprachmodelle unter denselben Bedingungen zu bewerten, nämlich um die Sprachmodelle zu vergleichen. Normalerweise gibt es viel mehr BLM Dokumente als TLM Dokumente. Aber hier verwenden wir die BLM Daten mit ähnlicher Größe als TLM Daten und diese BLM Daten werden auch entsprechend der Anzahl der TLM Dokumente aufgeteilt, um die Versuche anschaulicher durchzuführen. In der Abbildung 3.3 umfassen diese Trainingsdaten hier nicht nur alle Dokumente in einer Textform, die zum Aufbau der Sprachmodelle verwendet werden, sondern auch die separaten Dokumente, die nachher zum Aufbau des Klassifikationsmodells benutzt werden. Hier gibt es insgesamt 64 TLM Dokumente und die BLM Daten werden durchschnittlich in 65 Dokumente aufgeteilt.

Nach der Verarbeitung der BLM und TLM Dokumente können die Sprachmodelle aufgebaut werden. Hier wird der Befehl (3.1) verwendet. Das bedeutet, dass das Vokabular "en.lm.vocab" hier benutzt wird, um die Wahrscheinlichkeit der Wortfolgen zu berechnen. Die Wörter in dem Vokabular werden entsprechend ihrer Häufigkeiten in den BLM Trainingsdaten berücksichtigt, um die Wahrscheinlichkeit der Wortfolge zu berechnen. Die Wörter, die im Text, aber nicht im Wortschatz stehen, werden als **Out of Vocabulary(OOV)** angesehen. Alle Wörter dieses Typs werden als ein einheitlicher Typ OOV darstellt und deren Wahrscheinlichkeiten werden zur Wahrscheinlichkeit von OOV addiert. Für jedes Wort im Text werden höchstens die zwei vorherigen Wörter berücksichtigt. Hier wird die Smoothing Methode "- interpolate - kndiscount" benutzt, um das Data Sparsity Problem zu lösen, nämlich um die Möglichkeit der Wahrscheinlichkeit der Wortfolge gleich

”0” zu vermeiden. Die aufgebauten Sprachmodelle BLM und TLM werden im Prozess des Trainings von den unklassifizierten Dokumenten weiter verwendet.

$$\begin{aligned} ngram\text{-}count \quad -vocab \quad en.lm.vocab \quad -order \quad 3 \quad -text \quad BLMTraining \\ -lm \quad BLM \quad -interpolate \quad -kndiscount \end{aligned} \tag{3.1}$$

Nachdem die Sprachmodelle von BLM und TLM aufgebaut worden sind, sollten sie die Perplexität der Tuningdaten(Testdaten) bewerten. Hier wird die Perplexität der ”**BLM Tuningdaten/Testdaten**” auf Basis von dem originalen BLM berechnet. Analog wird die Perplexität von ”**TLM Tuningdaten/Testdaten**” mit dem originalen TLM berechnet. Diese berechneten Perplexitäten werden nachher mit den Perplexitäten der neuen Sprachmodelle verglichen, um zu schauen, ob die neu aufgebauten Sprachmodelle schon verbessert worden sind.

Mit dem Befehl (3.2) wird das **Klassifikationsmodell** auf Basis von den separaten Trainingsdateien mit dem Werkzeug **Rainbow** aufgebaut und dieses Modell wird für die Bewertung der Utterances der unklassifizierten Dokumente weiter verwendet.

$$\begin{aligned} rainbow \quad -d \quad \tilde{}/Klassifikationsmodell \quad --method = tfidf \\ --index \quad \tilde{}/Training\_sets/* \end{aligned} \tag{3.2}$$

3.2. Trainieren durch die unklassifizierten Dokumente

Das Hauptziel von dieser Arbeit ist, die Menge an Trainingsdaten von TLM und BLM durch Einlesen der unklassifizierten Dokumente zu vergrößern. Dazu werden die Dokumente auf der Utterance-Ebene berücksichtigt und die nützlichen Utterances werden als neue Trainingsdaten extrahiert. Hierfür wird jede Utterance mit einem ”Score” bewertet, um zu bestimmen, ob diese Utterance zu den Trainingsdaten hinzugefügt oder ignoriert werden kann. Die Formeln der Scores((3.3)und (3.4)) beschreiben, wie die Scores einer Utterance zu berechnen sind.

$$Score(T) = \frac{\frac{1}{P(utt|T)} * DW(T)}{\frac{1}{P(utt|T)} * DW(T) + \frac{1}{P(utt|B)} * DW(B)} \tag{3.3}$$

$$Score(B) = \frac{\frac{1}{P(utt|B)} * DW(B)}{\frac{1}{P(utt|T)} * DW(T) + \frac{1}{P(utt|B)} * DW(B)} \tag{3.4}$$

Wir können die Formel(3.3) als ein Beispiel nehmen, um detailliert zu erklären, wie die Utterances bewertet werden. Dabei bezeichnet P(utt|T) die berechnete Perplexität einer Utterance von dem themenspezifischen Sprachmodell. DW(T) stellt dar, wie wahrscheinlich ein Dokument, das die zu bewertende Utterance enthält, zu der Klasse Topic gehört. Die Formeln aus der Literatur [SGN05] werden wie im Folgenden angepasst. Eine vom themenspezifischen Sprachmodell bewertete Utterance ist themenbezogener, wenn ihre Perplexität niedriger ist. Ebenfalls spielt das Dokument, zu dem die Utterances gehören, eine Rolle. Je höher die Wahrscheinlichkeit ist, dass ein Dokument nach der Klassifikation zu der Klasse ”Topic” gehört, umso themenbezogener sind die Utterances in diesem Dokument. Deswegen ist Score(T) eine monotone und normierte Funktion, die einen Wert zwischen [0, 1] liefert. Er spiegelt die Korrelation einer Utterance mit dem themenspezifischen Sprachmodell wieder. Analog beschreibt die Funktion (3.4) den Zusammenhang von einer Utterance und dem Background Sprachmodell. Dadurch müssen zwei Programme (SRILM und Rainbow) verwendet werden, um die Perplexität und die Wahrscheinlichkeit der Klassifikation eines Dokuments zu bekommen.

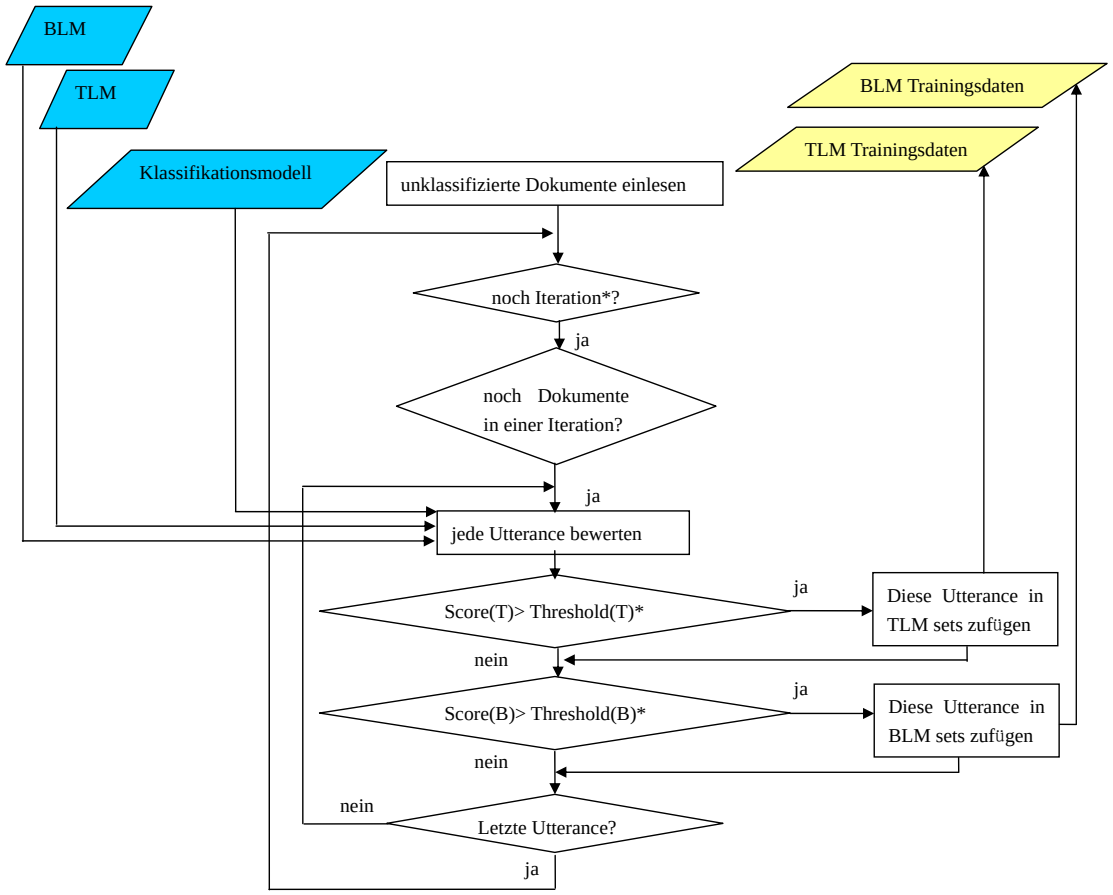


Abbildung 3.6.: Entwurf des Trainings für die unklassifizierten Dokumente

Die Abbildung 3.6 ist Teil von dem Ablaufdiagramm 3.2 und zeigt die Hauptaufgabe der ganzen Arbeit. In diesem Prozess werden die unklassifizierten Dokumente eingelesen. Die Dokumente werden nacheinander extrahiert, gesäubert und mit dem Werkzeug Rainbow erzeugten Klassifikationsmodell bewertet. Für jedes Dokument wird jede Utterance extrahiert, um ihre Perplexität mit dem schon aufgebauten Sprachmodell **TLM** und **BLM** zu berechnen. Dann hat jede Utterance zwei Scores: Score(T) entsprechend TLM und Score(B) entsprechend BLM. Wenn Score(T) größer als der Schwellwert **Threshold(T)** ist, dann bedeutet es, dass diese Utterance als neue Trainingsdatei von TLM extrahiert werden kann. Analog können die für BLM geeigneten Utterances auch gefunden. Die extrahierten Utterances werden neue Dokumente der Trainingsdaten aufbauen. Jedes neue Dokument beinhaltet 30 Utterances. Nach diesem Prozess werden die nützlichen Utterances aus den unklassifizierten Dokumenten extrahiert und dadurch die Trainingsdateien von TLM und BLM vermehrt.

3.3. Neues Aufbauen des Sprachmodells und die entsprechende Bewertung

Die Aufgabe dieses Teils ist auf Basis von den schon vermehrten Trainingsdateien die neuen Sprachmodelle aufzubauen und ihren Perplexitäten zu berechnen. Der detaillierte Prozess wird in der Abbildung 3.7 beschrieben. Hierfür werden zuerst die neuen Sprachmodelle **BLM** und **TLM** mit den aktualisierten **BLM Trainingsdaten** und **TLM Trainingsdaten** aufgebaut. Diese aus den schon vermehrten Trainingsdaten aufgebauten Sprachmodelle können leider nicht direkt benutzt werden. Solche neu erhaltenen Trainingsdaten

haben standardmäßig die gleichen Gewichte wie die originalen Trainingsdaten, obwohl die Qualität der neuen Daten schlechter als die der originalen Daten ist. Deswegen wird das neu aufgebauten Sprachmodell schlechter als das originalen Sprachmodell, wenn die Gewichte nicht neu berechnet worden sind. Deswegen muss ein neues Sprachmodell in das originale Sprachmodell mit den entsprechenden Gewichten integriert werden, um ein verbessertes Sprachmodell zu erstellen.

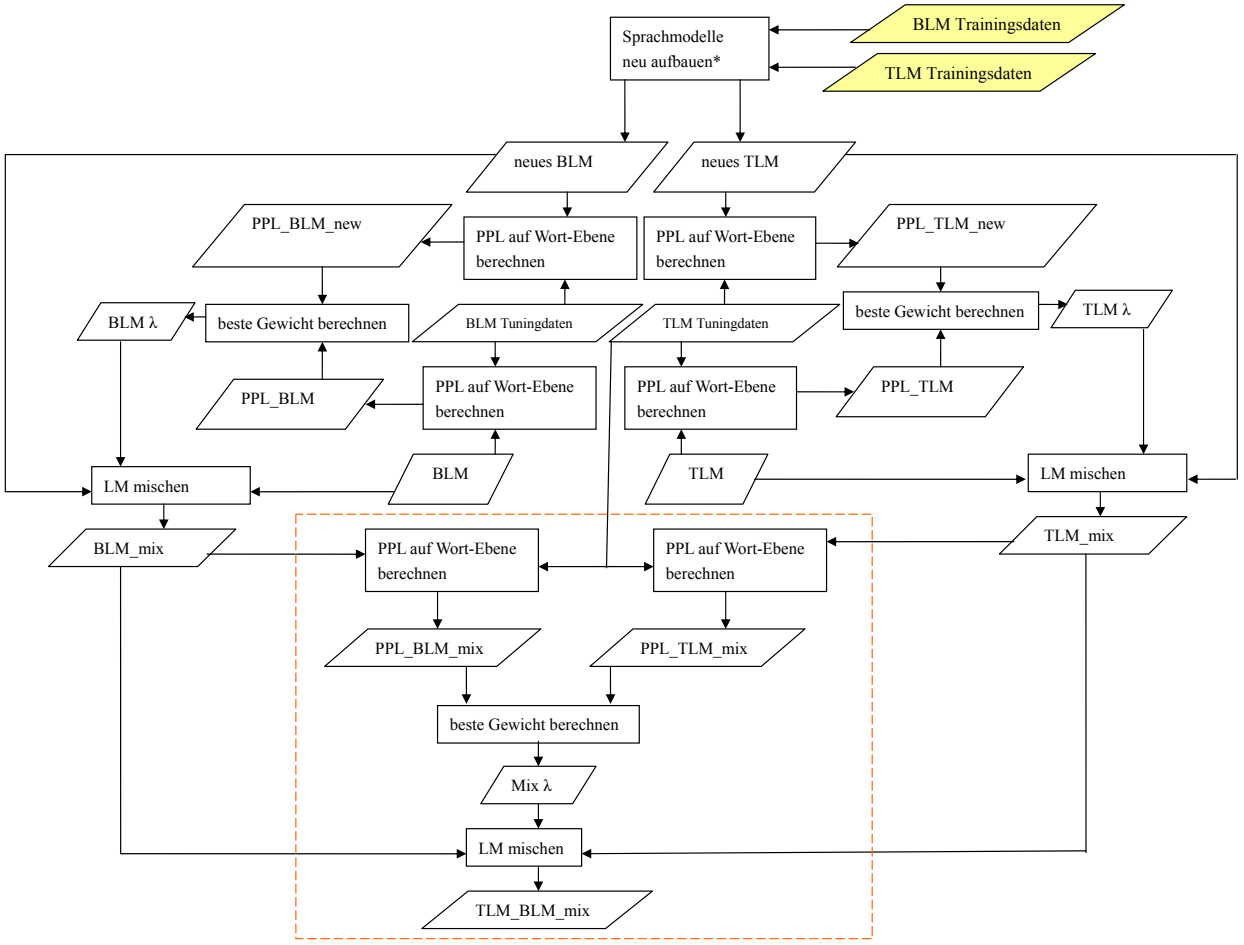


Abbildung 3.7.: Entwurf vom neuen Aufbauen und Testen des Sprachmodells

Hierdurch müssen neues und originales Sprachmodell gemäß den entsprechenden Tuningdaten oder Testdaten die Perplexität auf Wort-Ebene berechnen. Betrachten wir **TLM** als ein Beispiel. Mit den Befehlen 3.5 wird die Datei **TLMTest.txt** auf der Wort-Ebene mit dem originalen Sprachmodell **TLM** und dem neuen Sprachmodell **TLM_new** analysiert. Die Ergebnisse werden entsprechend unter **PPL_TLM** und **PPL_TLM_new** speichert. Diese zwei Dokumente werden mit einem Skript "**compute-best-mix**" von SRILM verwendet, um das beste Gewicht **TLM** λ zu berechnen, mit dem die Sprachmodelle **TLM** und **TLM_new** optimal zu einem neuen Sprachmodell **TLM_mix** interpoliert werden können. Das Gewicht **BLM** λ kann ebenfalls auf Basis von den mit **BLM Tuningdaten/Testdaten** bewerteten Ergebnissen berechnet werden. Mit diesem Gewicht, dem originalen Sprachmodell und dem neuem Sprachmodell von BLM wird eines neue Sprachmodell **BLM_mix** optimal aufgebaut.

$$\begin{aligned} ngram\ \ -debug\ \ 2\ \ -lm\ \ TLM\ \ -ppl\ \ TLMTest.txt > PPL\_TLM \\ ngram\ \ -debug\ \ 2\ \ -lm\ \ TLM\_new\ \ -ppl\ \ TLMTest.txt > PPL\_TLM\_new \end{aligned} \tag{3.5}$$

In den Versuchen, die Parameter probieren und festzulegen, werden die Tuningdaten von TLM und BLM verwendet und die Trainingsdateien von TLM und BLM werden jeweils vermehrt. Dadurch können TLM und BLM entsprechend verbessert werden. Wie vorher gesagt, kann ein Background Sprachmodell zwar die Informationen eines spezifischen Themas nicht liefern, aber es kann die Warhscheinlichkeitsverteilung der Wortfolge liefern, die die Struktur der Phrasen oder der Sätze widerspiegelt. Deshalb kann ein verbessertes Background Sprachmodell auch dabei positiv helfen, die Qualität eines Topic Sprachmodells zu verbessern. Aus diesem Grund können die schon verbesserten TLM und BLM mit dem Skript **compute-best-mix** nochmal in ein einheitliches Sprachmodell integriert werden, um das TLM Sprachmodell zu verbessern. Das entsprechende Gewicht für die Interpolation wird durch die Ergebnisse, die von **TLM_mix** und **BLM_mix** auf der Wort-Ebene basierend auf **TLM Testdaten** bewertet werden, berechnet. Diese Interpolation wird nur im Evaluierungsversuch (siehe Kapitel 4.4 und 5.4) mit den schon festgelegten Parametern zusätzlich gemacht, um das letztlich verbesserte themenspezifische Sprachmodell **TLM_BLM_mix** zu bekommen und die entsprechende Qualität zu bewerten. Wie am Anfang dieses Kapitel gesagt, werden in einer solchen Evaluation die Testdaten von BLM und TLM verwendet.

Zur Interpolation der Sprachmodelle müssen die Tuningdaten oder Testdateien mit den Sprachmodellen auf der Wort-Ebene bewertet werden, um das Gewicht der Interpolation zu erhalten. Aber um zu testen, ob die Qualität eines Sprachmodells schon verbessert wird, müssen sie auf der Dokument-Ebene auswertet werden. Die Qualität der aufgebauten Sprachmodelle **TLM_mix**, **BLM_mix** und **TLM_BLM_mix** sollten mit dem Befehl "ngram -debug 0" nämlich auf der Dokument-Ebene mit den entsprechenden Testdateien ausgewertet werden. Dadurch kann gesehen werden, ob ein Sprachmodell schon nach der Voradaption der neuen Trainingsdateien verbessert worden ist.

4. Versuchsaufbau

Im Kapitel 3 ist die Idee abklärt worden, wie die neuen Trainingsdateien aus den umfangreichen unklassifizierten Dokumenten extrahiert werden können. Für diesen Entwurf müssen einige Parameter berücksichtigt werden, um deren Leistung am besten einzubringen. Im Prozess des Einlesens der neuen Dokumente muss entschieden werden, welchen Schwellwert der Score(T) 3.3 oder Score(B) 3.4 überschreiten muss, so dass die Utterances in die Trainingsdateien hinzugefügt werden können. Hierfür müssen die Schwellwerte (Thresholds) als Parameter durch die Versuche bestimmt werden. In gleicher Weise kann der Parameter "Anzahl der Dokumente" nach den Versuchen des Programms festgelegt werden, der beschreibt, nach wie vielen Dokumenten das aktualisierte Sprachmodell zur Bewertung der Utterances benutzt wird. Der Parameter "Anzahl der Iterationen" bestimmt, nach wievielen Iterationen das Klassifikationsmodell neu aufgebaut und zur Bewertung verwendet wird. Für jeden Versuch ändert sich nur ein Parameter und die anderen Parameter werden mit den entsprechenden Werten festgelegt. Unter dieser Bedingung kann gesehen werden, ob ein Parameter eine Rolle spielen kann, so dass die Sprachmodelle sich verbessern können. Durch diese Versuche werden die am meisten geeigneten Werte für die jeweiligen Parameter gefunden.

Da es umfangreiche unklassifizierte Dokumente gibt, muss die Anzahl der einzulesenden Dokumente bestimmt werden, um die Zeitkosten zu beschränken. Die Abbildung 4.1 zeigt die Zeitkosten der Versuche mit neuem Aufbauen des Sprachmodells. Diese Versuche wurden auf einem Rechner, der zwei CPU(2.33 GHZ) und 2048 M Memory hat, durchlaufen. Aus der Abbildung kann gesehen werden, dass sich die Zeitkosten mit dem Einlesen der Dokumente exponentiell erhöhen. Die maximalen Zeitkosten eines Versuchs, in dem 20000 unklassifizierte Dokumente eingelesen werden, betragen fast 180000 Sekunden (50 Stunden). Je mehr einzulesende Dokumente es gibt, desto größer sind die aufgebauten Sprachmodelle und die aufgebauten Klassifikationsmodelle. Die Bewertungen der Utterances mit den größeren Sprachmodellen und dem größeren Klassifikationsmodell können zu erhöhten Zeitkosten führen, da für jede Utterance eine Bewertung mit den Sprachmodellen durchgeführt wird. Inklusive anderer Versuche wie solche, mit denen die Schwellwerte aus ein paar Werten bestimmt werden, wird die Anzahl der Versuche mehr als 20 betragen. Aus diesen Gründen werden für alle Tuningversuche 20000 unklassifizierte Dokumente eingelesen. Für die Evaluierungsversuche werden 50000 Dokumente eingelesen, um anschaulichere Evaluationsergebnisse zu erhalten und damit eine triftige Evaluation der Methode dieser Arbeit erhalten werden kann.

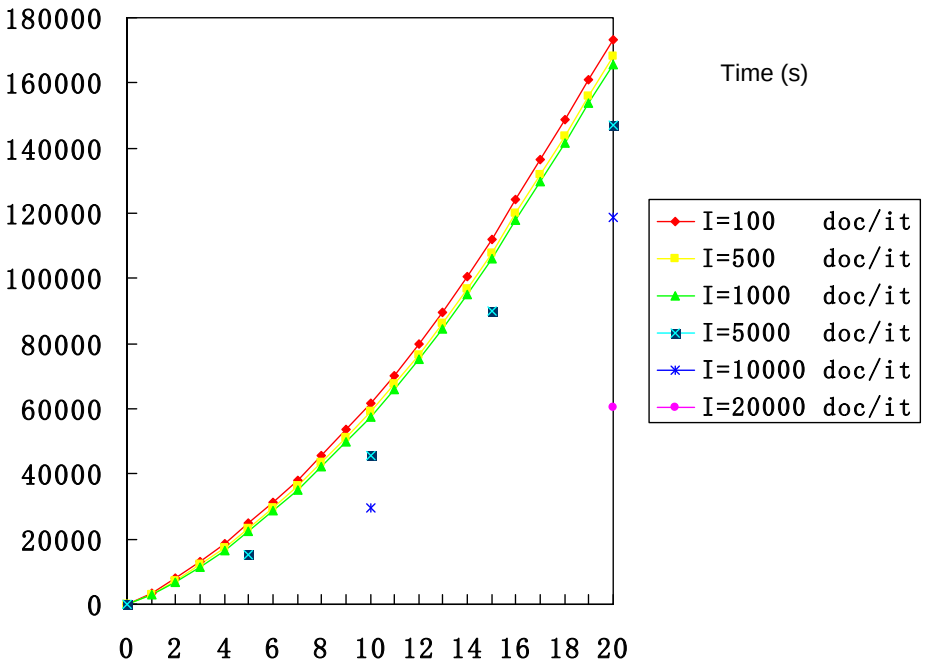


Abbildung 4.1.: Zeitkosten für die Versuche

4.1. Experimente mit neuem Aufbauen des Sprachmodells

Im Prozess des Trainierens der unklassifizierten Dokumente können die Sprachmodelle BLM und TLM verbessert werden, nachdem die nützlichen Utterances akzeptiert worden sind. Für die Bewertung einer Utterance spielt ihre Perplexität eine Rolle. Abhängig von den Formeln Score(T) 3.3 und Score(B) 3.4 kann gesehen werden, dass die Scores beeinflußt werden, wenn sich die Perplexitäten geändert haben. Eine Perplexität wird durch ein Sprachmodell bewertet. In diesem Fall kann ein Sprachmodell mit besserer Qualität eine genauere Perplexität liefern. Damit kann man darüber nachdenken, ob es möglich ist, dass die mit dem neu aufgebauten Sprachmodell akzeptierten Utterances bessere Qualitäten im Vergleich mit solchen aus dem originalen Sprachmodell bekommenden Utterances erhalten können. Dadurch werden das themenspezifische Sprachmodell und das Background Sprachmodell nach einer bestimmtem Anzahl der eingelesenen Dokumente neu aufgebaut. Diese Idee wird mit ein paar Werte getestet, um zu schauen, nach wie vielen eingelesenen Dokumente das letztlich aufgebaute Sprachmodell die beste Leistung erhalten kann.

Für jeden getesteten Wert in diesen Versuchen werden 20000 Dokumente eingelesen und die Anzahl der in der jeweiligen Iteration erhaltenden Dokumente werden zwischen 100, 500, 1000, 5000, 10000 und 20000 ausgewählt. Im Folgenden bezeichnet ”I” den Parameter, der bestimmt, nach wie vielen Dokumenten die Sprachmodelle neu aufgebaut werden. Dabei bezeichnet I = 100, dass die Sprachmodelle TLM und BLM neu aufgebaut werden, nachdem je 100 neue Dokumente eingelesen worden sind. Die neu aufgebauten Sprachmodelle werden sofort verwendet, um die Scores der nachher neu ankommenden Utterances zu berechnen. In diesem Beispiel werden die ersten 100 neuen Dokumente abhängig von den originalen Sprachmodellen zunächst bewertet, dann werden die neue Sprachmodelle TLM und BLM mit den schon vermehrten Trainingsdaten aufgebaut. Danach werden die Perplexitäten für die nachher ankommenden 100 Dokumente mit diesen neuen Sprachmodellen bewertet und weitergehend werden die Scores berechnet. Mit diesen Scores kann entschieden werden, welche Utterances in die Trainingsdaten hinzugefügt werden können. Mit diesem iterativen Verfahren (nämlich nach 200 Dokumentengruppen) können die letzt-

lichen Sprachmodelle TLM und BLM erhalten werden. Von diesen Sprachmodelle können die entsprechenden Tuningdaten bewertet werden, um zu schauen, ob die Qualität der Sprachmodelle nach dem Trainieren sich schon verbessert hat. Die Qualität kann durch die Perplexität gesehen werden, welche die Tuningdaten auf Dokument-Ebene auswertet. In gleicher Weise hat der Versuch mit I = 500 insgesamt 40 Dokumentengruppen, wobei die Sprachmodelle insgesamt 40 mal neu aufgebaut werden. Analog werden die Sprachmodelle 20 mal neu aufgebaut für I = 1000 und sie werden 4 mal für I = 5000 aufgebaut. Für I = 10000 werden die Sprachmodelle 2 mal neu aufgebaut. Es bedeutet für I = 20000, dass die ankommenden Dokumente vom Anfang bis Ende nur mit den originalen Sprachmodellen bewertet werden. Es bedeutet nämlich für die Bedingung I = 20000, dass die Sprachmodelle während dem Prozess des Trainierens durch Dokumente nur einmal neu aufgebaut werden und die extrahierenden Utterances nicht von diesem Parameter beeinflusst worden sind. Die anderen Parameter werden als bestimmte Werte gesetzt. Dazu werden die Thresholds von TLM und BLM als 0.5 angenommen und diese Versuche sind ohne neues Aufbauen des Klassifikationsmodells. Die Ergebnisse dieser Versuche werden im Kapitel 5.1 weiter demonstriert und analysiert.

4.2. Versuche mit den varianten Threshhold

Für die ganzen Arbeit sind die Formeln von Score 3.3 und 3.4 die wichtigst. Sie stellen dar, wie eine Utterance bewertet werden kann. Nach dieser Bewertung muss man noch überlegen, welche Schwellwerte hier ausgewählt werden können, um die wahrscheinlichst themenbezogenen Utterances zu extrahieren, so dass die Qualität der Sprachmodelle am meistens verbessert werden können. Zweifellos ist der Parameter Threshhold der wichtigst in den Versuche aller Parameter.

Hier werden 20000 Dokumente für jeweiligen versuchten Threshhold eingelesen, wobei je 2000 Dokumente als eine Iteration gesehen werden und es insgesamt 10 Mals neues Aufbauen der Sprachmodelle gibt. Die Dokumente werden jedesmal von dem gerade neu aufgebauten Sprachmodell bewertet und weitergehend werden die Score davon berechnet. Die Formeln 3.3 und 3.4 sind normiert und deswegen sind zwischen [0,1]. Außerdem sind diese Formeln monoton, damit können einen besten Schwellwert gefunden werden. Hierfür können die Threshholds zuerst von 0.1 bis 0.9 mit der Intervall 0.1 genommen werden, um zu finden, unter welchem Wert ein Sprachmodell am meisten verbessert werden kann. Ein Schwellwert ist dafür besser, wenn die unter diesem Schwellwert bewertete Perplexität niedriger ist. Nachdem ein Schwellwert gefunden worden ist, können einige Versuche weiter gestellt werden, so dass die präzisere Schwellwerte gefunden werden können.

4.3. Experimente mit neuem Aufbauen des Klassifikationsmodells

Im Prozess des Einlesens neuer Dokumente werden die Utterances extrahiert, die relevant für das Sprachmodell TLM oder das Sprachmodell BLM sind. Sie werden zu **TLM Trainingsdaten** oder **BLM Trainingsdaten** nicht nur in Textform, sondern auch in die separaten Trainingsdateien als neue Files hinzugefügt. Dazu dienen die **TLM Trainingsdaten** und die **BLM Trainingsdaten** als Quelle um die entsprechenden Sprachmodelle aufzubauen. Die separaten Files dienen als Quelle der Trainingsdateien der Textklassifikation, um das Klassifikationsmodell aufzubauen. Durch die Verarbeitung von diesem Klassifikationsmodell kann für ein Dokument entschieden werden, ob es zu der Klasse Topic oder der Klasse Background gehört.

In diesem Fall kommt es zur Frage, ob der verbesserte Textklassifier als Folge von der Zunahme der Trainingsdateien die Dokumentenwahrscheinlichkeiten **DW(T)** und **DW(B)**

für ein Dokument genauer berechnen und weitergehend die neu eingelesenen Dokumente effizienter extrahieren kann. Basierend auf dieser Frage werden einige Versuche durchlaufen. Aus diesen Versuche kann bestimmt werden, ob das neue Aufbauen des Klassifikationsmodells einhergehend mit der Zunahme der Trainingsdateien für dieses Klassifikationsmodell dabei helfen wird, die besseren Utterances zu extrahieren. Durch diese Versuche kann bestimmt werden, ob das neue Aufbauen des Klassifikationsmodells ein Faktor für die Verbesserung des letztlich erhaltenen Sprachmodells ist.

Hierfür werden 20000 Dokumente eingelesen, dabei werden je 1000 Dokumente als eine Iteration gesehen und es gibt insgesamt 20 mal ein neues Aufbauen der Sprachmodelle . Die Dokumente werden jedes mal von dem gerade neu aufgebauten Sprachmodell bewertet und weitergehend werden die Scores davon berechnet. Die Schwellwerte von TLM und BLM werden hier als 0.5 angenommen.

In diesen Versuchen wird das Klassifikationsmodell nach jeweils 1, 2, 3, 4, oder 5 Iterationen neu aufgebaut. Damit wird dieses neu aufgebaute Modell verwendet, das **DW(T)** und **DW(B)** der kommenden Dokumente aktualisiert. Hier werden die Sprachmodelle TLM und BLM noch nach jeweils 1000 Dokumenten neu aufgebaut und deren Perplexitäten werden entsprechend bewertet. Für jeden Versuch werden 20 Perplexitäten von TLM und BLM erhalten, die durch die Bewertung derselben Tuningdaten von den aktualisierten Sprachmodellen und dem Klassifikationsmodell berechnet werden. Aus diesen Perplexitäten kann gesehen werden, wie oft ein Klassifikationsmodell neu aufgebaut wird, um die Sprachmodelle TLM und BLM am meisten zu verbessern. Es kann sein, dass manche Perplexitäten für unterschiedliche Iterationsnummern in den Versuchen manchmal gleich sind, weil die Division mit unterschiedlicher Anzahl den gleichen Rest erhalten kann. Man sieht es zum Beispiel, dass die Ergebnisse der dritten Iteration für Iterationsnummer 4 und 5 gleich sind, da bis dahin die beiden Klassifikationsmodelle noch nicht neu aufgebaut worden sind, das heißt, dass die Utterances noch mit dem originalen Klassifikationsmodell bewertet wurden. Aber um den besten Wert zu finden braucht man nur die letztlichen Ergebnisse zu vergleichen. Hier bezeichnet die Iterationsnummer, nach wie vielen Iterationen bzw. nach wie vielem neuen Aufbauen der Sprachmodelle das Klassifikationsmodell neu aufgebaut wird.

4.4. Evaluierungsversuche mit den schon bestimmten Parameter

Aus obigen Versuchen wurden einige Parameter getestet. Dadurch wurden die besten Werte gefunden, die die beste Leistung des Programms liefern können. Hier wird die Anzahl benutzt, nach wie vielmaligen Einlesen der Dokumente die neu aufgebauten Sprachmodelle TLM und BLM zur Bewertung der Perplexitäten der neu einlesenden Dokumente angeboten werden können. Aus den Versuchen wurden auch die passendsten Schwellwerte für das Programm gefunden, über welche die Utterances entsprechend als themenbezogen mit TLM oder BLM akzeptiert werden. Außerdem wurde getestet, ob außer dem neuen Aufbauen der Sprachmodelle das neue Aufbauen des Klassifikationsmodells eine Rolle dafür spielen kann, das heißt, ob dieses Programm mit regelmäßigem neuen Aufbauen des Klassifikationsmodells letztlich bessere Sprachmodelle aufbauen kann und nach wie viele Iterationen dies gemacht werden kann.

Diese gefundenen Parameter werden in den Evaluierungsversuchen verwendet. Dazu werden die identische Testdaten für diese Versuche benutzt. Hierfür werden 50000 Dokumente eingelesen und jeweils 1000 Dokumente werden als eine Iteration angesehen. Damit gibt es insgesamt 50 Iterationen für jeden Versuch. Um zu schauen, ob die in der Arbeit verwendete Methode das Sprachmodell für das Thema schließlich verbessert, können hier 3

Versuche durchgeführt werden. In diesen Versuchen werden die Testdaten mit der Baseline Methode, nur mit dem verbesserten Sprachmodell TLM und mit einem von den verbesserten Sprachmodellen TLM und BLM interpolierten Sprachmodell bewertet. Diese Sprachmodelle werden während des Einlesens der neuen Dokumente aktualisiert. Nach dem jeweiligen Einlesen von 1000 Dokumenten werden die Perplexitäten für diese drei Sprachmodelle mit den identischen Testdaten neu bewertet.

Der Versuch Baseline bedeutet, dass alle vorhandenen Daten als Trainingsdaten des themenspezifischen Sprachmodells angesehen werden können. Diese Daten beinhalten die Trainingsdaten vom themenspezifischen Sprachmodell und vom Background Sprachmodell, und nachher werden sie auch die eingelesenen Dokumente beinhalten. Zuerst werden diese Daten in einen vereinigten Text kopiert und basierend auf diesem Text wird ein Sprachmodell Baseline aufgebaut. Danach werden für dieses Sprachmodell alle Utterances der gerade eingelesenen Dokumente akzeptiert und zu den Baseline Trainingsdaten hinzugefügt. Nachdem jeweils 1000 Dokumente gelesen worden sind, wird das Sprachmodell Baseline neu aufgebaut und die Perplexität auswertet.

Während des Verlaufs dieses Programms wird das themenspezifischen Sprachmodell verbessert werden. Der zweite Versuch berücksichtigt nur die Verbesserung des themenspezifischen Sprachmodells. Hierfür werden nur die **TLM Trainingsdaten** benutzt, um eines Sprachmodell TLM aufzubauen. Durch das Einlesen der unklassifizierten Dokumente werden nur die "nützlichen" Utterances als neu hinzugefügte Trainingsdaten extrahiert, wobei "nützlich" bedeutet, dass die Scores einer Utterance den in obigen Versuchen schon festgelegten TLM Schwellwert überschreiten können. Für jede Iteration wird dieses Sprachmodell abhängig von den aktualisierten Trainingsdaten neu aufgebaut und seine Perplexität wird bewertet.

Wie im Kapitel 3.3 erwähnt, kann eines Background Sprachmodell auch dabei helfen, die Qualität eines themenspezifischen Sprachmodells zu verbessern. Im Prozess des Programms werden nicht nur das themenspezifischen Sprachmodell sondern auch das Background Sprachmodell verbessert. Deswegen wird versucht, dass nach jeweils 1000 Dokumenten ein integriertes Sprachmodell neu aufgebaut wird, das von solchen schon verbesserte TLM und BLM interpoliert wird. Dieses integrierte Sprachmodell kann ganz wahrscheinlich dadurch verbessert werden, da TLM und BLM schon die verbesserten Sprachmodelle sind.

In diesen Evaluierungsversuchen werden die drei genannten Sprachmodelle die Testdaten von TLM bewerten. Durch die Perplexitäten, die jeweils nach dem Einlesen von jeweils 1000 Dokumenten berechnet werden, kann gesehen werden, welche Methode sich am besten auswirken kann. Dadurch können wir ein relativ gutes themenspezifisches Sprachmodell bekommen und es kann auch gesehen werden, wie viel das themenspezifischen Sprachmodell mit der Methode in dieser Arbeit verbessert werden kann.

5. Ergebnisse

In diesem Kapitel werden die Ergebnisse der Versuche im Kapitel 4 angegeben. Durch die Analyse und die Erklärung wird gesehen, welche Varianten sinnvoll sind und wie die entsprechende Werte sein sollten. Letztlich in diesem Kapitel wird eine Tabelle ausgegeben, um zu schauen, wie dieses Programm nach dem Verlauf das themenspezifische Sprachmodell verbessern kann. Dieses Kapitel wird entsprechend dem Kapitel 4 strukturiert.

5.1. Evaluation der Versuche mit neuem Aufbauen der Sprachmodelle

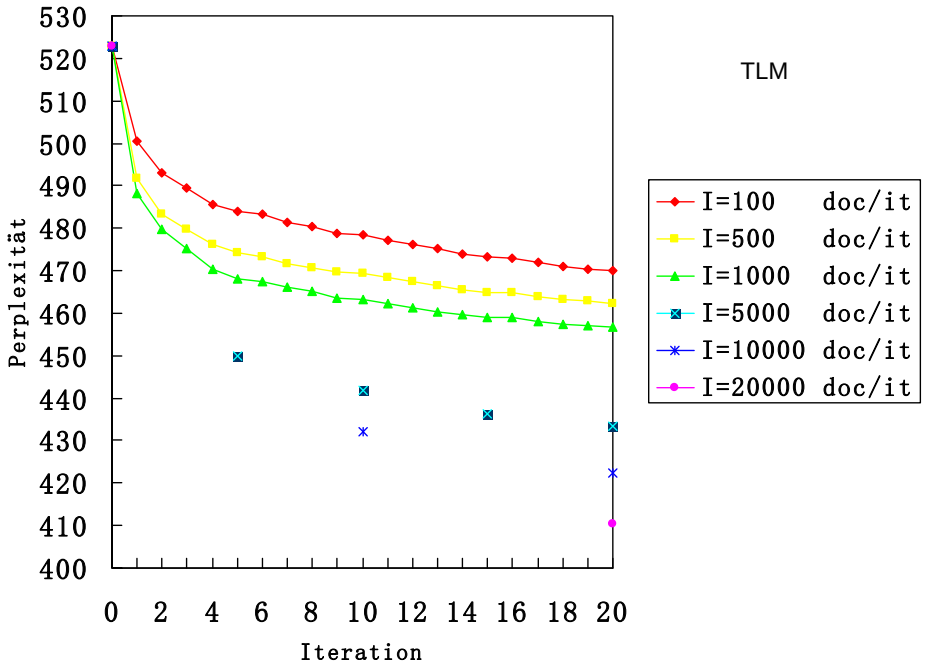


Abbildung 5.1.: Ergebnisse von TLM mit neuem Aufbauen der Sprachmodelle

In diesem Teil werden die Ergebnisse der Versuche im Unterkapitel 4.1 angegeben und analysiert. Es zeigt in der Abbildung 5.1 die Perplexitäten der themenspezifischen Sprachmodelle, die nach jeweils einer bestimmten Anzahl an Dokumenten neu aufgebaut wurden.

Damit kann zuerst gesehen werden, dass in jedem Versuch die Perplexität des themenspezifischen Sprachmodells während des Verlaufs des Programms immer abnimmt. Das bedeutet, dass die Sprachmodelle in allen Versuchen während des Einlesens der neuen Dokumente immer verbessert wurden. Außerdem kann man sehen, dass der Versuch mit **I = 20000** letztlich die niedrigste Perplexität erzielt bzw. die beste Leistung geliefert hat. Das bedeutet, dass der Versuch das beste themenspezifische Sprachmodell anbieten kann, wenn es immer mit den originalen Trainingsdaten die Utterances bewertet. Die Abbildung

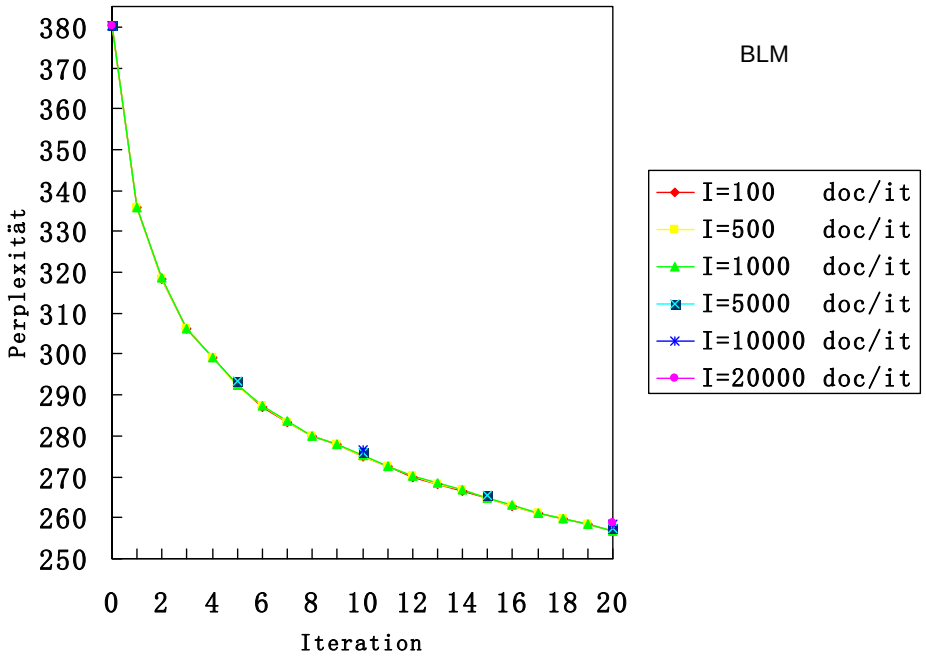


Abbildung 5.2.: Ergebnisse von BLM mit neuem Aufbauen der Sprachmodelle

5.2 zeigt die Ergebnisse des Background Sprachmodells in den Versuchen. Dabei kann gesehen werden, dass die bewerteten Perplexitäten aller Background Sprachmodelle durch den Verlauf des Programms stark gefallen waren. Das heißt, dass alle Background Sprachmodelle in diesen Versuchen viel verbessert wurden, trotzdem sie mit unterschiedlichen Parameter waren. Aber im Gegensatz zur Abbildung TLM (5.1) ist hier nicht das Background Sprachmodell mit der Variante **I = 20000** das Beste. Deswegen wurde hier ein neu interpoliertes Sprachmodell für jeden Versuch aufgebaut, das von in jedem Versuch letztlich bekommenden TLM und BLM integriert wurde. Danach wurden diese interpolierten Sprachmodelle mit den TLM Tuningdaten bewertet, um zu sehen, unter welchem Wert sich das Programm am besten auswirkt bzw. das beste Sprachmodell anbieten kann. Im Anhang A.1 zeigt es alle entsprechenden Versuchsergebnisse und aus diesen Daten kann man sehen, dass für **I = 20000** das Programm das beste themenspezifische Sprachmodell liefern kann. Das bedeutet, dass hier die Utterances nur mit den originalen Trainingsdaten von TLM und BLM bewertet werden sollten, um die beste Leistung des Programms zu erreichen.

Jetzt werden diese Ergebnisse und die entsprechenden Ursachen analysiert. Nachdem die Sprachmodelle mehr nützliche Daten bekommen haben, können diese Sprachmodelle mit den aktualisierten Trainingsdaten neu aufgebaut werden und sie können bessere Qualität erhalten. Normalerweise können die Scores Score(T) und Score(B) einer Utterance mit diesen besseren Sprachmodellen genauer berechnet werden. Aber die Ergebnisse der Versuche zeigen, dass trotz der genaueren Bewertung der Utterances die neu akzeptierten Utterances

zu einer schlechten Leistung geführt haben. Es kann sein, dass die neuen Sprachmodelle zwar mehr themenbezogenen Daten beinhaltet haben, aber diese Daten sind toleranter als solche originalen Trainingsdaten. Unter dieser Bedingung ist es wahrscheinlich, dass die Qualität der mit diesem toleranteren Sprachmodells neu akzeptierten Utterances nicht so gut ist als solche, die nur von dem originalen Sprachmodell bewertet werden. Aus den bewerteten Perplexitäten von TLM und BLM kann man sehen, dass die Qualität von BLM besser als die von TLM ist. Deswegen kann BLM im Verlauf des Programms mehr als TLM verbessert werden. Diese gute Qualität kann eine Ursache sein, dass BLM mit dem neuen Aufbauen eine bessere Leistung erhalten kann. In diesem Fall spielt die Qualität der originalen Trainingsdaten eine mögliche Rolle. Aber die Ergebnisse von BLM sind miteinander so ähnlich, dass man nicht genau sagen kann, ob diese erhaltenen Ergebnisse zufällig sind oder ob die Qualität der originalen Trainingsdaten eines Sprachmodells wirklich die Ergebnisse mit neuem Aufbauen des Sprachmodells beeinflussen kann. Auch wenn die Qualität der originalen Trainingsdaten hierfür eine Rolle spielen kann, kann die Qualität des beliebig gegebenen themenspezifischen Sprachmodells ganz ohne Gewähr sein, so dass dieser Parameter Iterationsnummer, mit welchem die Sprachmodelle neu aufgebaut werden, ignoriert werden muss.

5.2. Ergebnisse der Versuche nach dem Parameter Threshold

Hier werden die Ergebnisse der Versuche im Unterkapitel 4.2 demonstriert. Die Abbildung 5.3 zeigt die Ergebnisse der Versuche, in welchen der Parameter Schwellwert jeweils von 0.1 bis 0.9 angenommen wurden. Aus dieser Abbildung kann man wissen, dass die letztlich

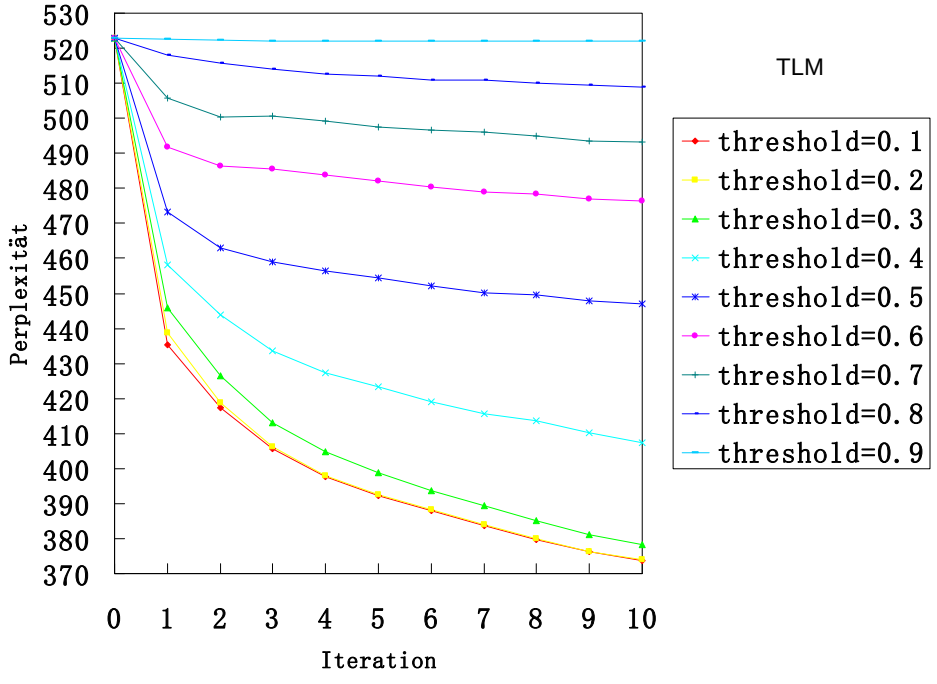


Abbildung 5.3.: Versuchsergebnisse nach den Thresholds von TLM

bewerteten Perplexitäten jedes Versuchs schon niedriger als solche vor dem Programmverlauf sind. Das bedeutet, dass die Qualität eines Sprachmodells sowieso verbessert wird, egal welchen Wert der Schwellwert angenommen hat. Aber wir können auch aus den Daten in der Tabelle A.1 sehen, dass die Zwischenergebnisse für die Schwellwerte 0.7 und 0.9 folgend mit den Iterationen im Verlauf nicht immer wieder verbessert wurden. Es kann davon

verursacht werden, dass innerhalb einer Iteration ganz wenige gelesenen Dokumente diese Schwellwerte überschritten haben und sie zufällig keine gute Qualität hatten. Deswegen hatten die Versuchsergebnisse mit den hohen Schwellwerten sich manchmal verschlechtert. Außerdem sind die Tendenzen aller Versuche untereinander sehr unterschiedlich. Dazu ist die Kurve vom Schwellwert 0.1 die steilste. Sie erzielte auch die niedrigste bzw. die beste Perplexität. Deswegen wird der Schwellwert des themenspezifischen Sprachmodells als 0.1 angenommen, weil mit ihm die beste Qualität des themenspezifischen Sprachmodells erzielt wurde.

Im Verlauf des Suchens nach dem besten Schwellwert des themenspezifischen Sprachmodells werden die Versuchsergebnisse des Background Sprachmodells gleichzeitig auch erhalten. In der Abbildung 5.4 werden sie gezeigt. Hier haben alle Versuche die Tendenz nach unten. Alle Zwischenergebnisse eines Versuchs stiegen auch nacheinander ab. Aus den Kurven und mit der Hilfe der Ergebnistabellen kann man sehen, dass der am besten geeignete Schwellwert 0.4 ist.

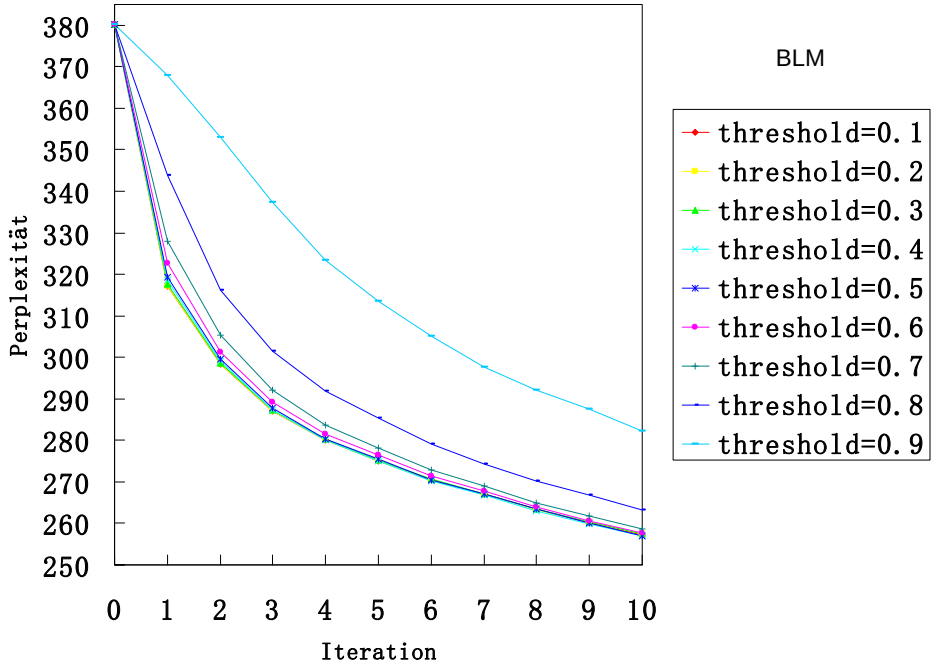


Abbildung 5.4.: Versuchsergebnisse nach den Thresholds von BLM

Um eine bessere Leistung des Programms zu erzielen, werden weitere Versuche durchgeführt, nachdem die besten Schwellwerte der Sprachmodelle TLM und BLM mit dem Intervall 0.1 im Bereich von 0.1 bis 0.9 gefunden worden sind. Aufgrund des Sparens des Zeitaufwands werden für das themenspezifischen Sprachmodell nur die an den Schwellwert 0.1 angrenzenden Werte 0.05 und 0.15 genommen und die entsprechenden Versuche werden durchgeführt. Für das Background Sprachmodell werden die direkt an 0.4 angrenzenden Schwellwerte 0.35 und 0.45 genommen. Das Ziel dieser Versuche liegt darin, dass die genaueren Schwellwerte gefunden werden können, um die Sprachmodelle mit besserer Qualität zu erhalten. Die Abbildung 5.5 beschreibt das genauere Testen für die Schwellwerte des themenspezifischen Sprachmodells. Darin kann man sehen, dass zwar die bewertete Perplexität jedes Versuchs vom Schwellwert 0.05, 0.10 und 0.15 offensichtlich niedriger als solche von den anderen Schwellwerte ist, aber es einen sehr kleinen Unterschied untereinander gibt. Deswegen wird der beste Schwellwert erst aus der Tabelle A.5 gefunden. Hier ist die zuletzt erhaltene Perplexität vom Wert 0.05 trotzt des kleinen Unterschieds

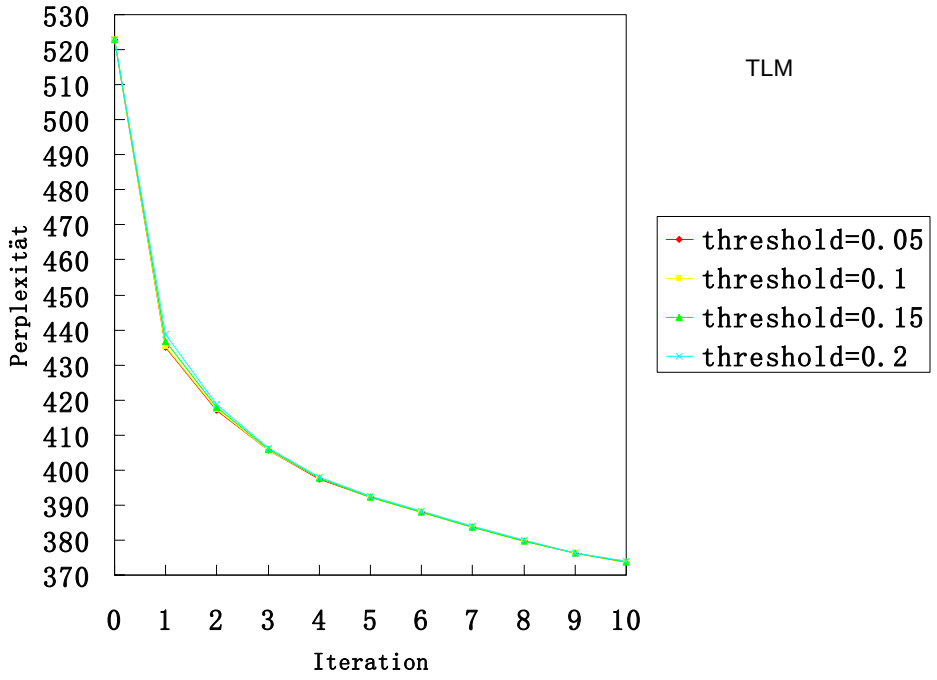


Abbildung 5.5.: Versuchsergebnisse nach detailierten Thresholds von TLM

die niedrigste. Dann wird in weiteren Versuchen der Schwellwert des themenspezifischen Sprachmodells als 0.05 angenommen. Analog ist es auch schwer aus der Abbildung 5.6

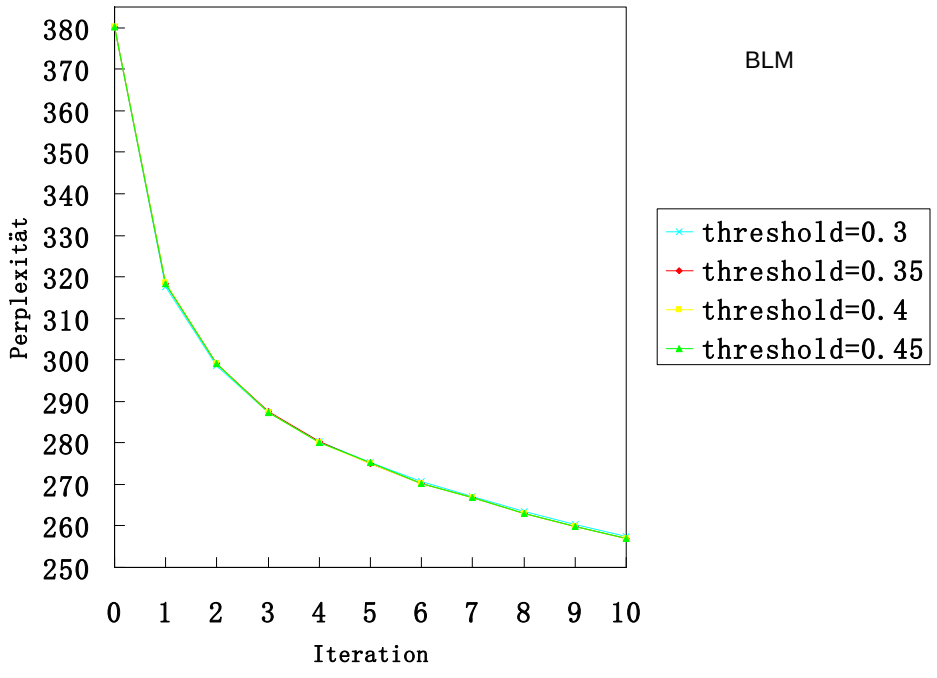


Abbildung 5.6.: Versuchsergebnisse nach den detailierten Thresholds von BLM

den besten Schwellwert vom Background Sprachmodell zu finden, da diese Versuchsergebnisse ganz nah aneinander gelegen haben. Mit der Hilfe der Tabelle A.6 kann der beste Schwellwert 0.35 gefunden, da mit ihm innerhalb dieser Versuche die beste schließlich be-

kommende Perplexität erzielt wird. Dann wird in weiteren Versuchen der Schwellwert 0.35 für das Background Sprachmodells benutzt.

5.3. Experimente mit neuem Aufbauen des Klassifikationsmodells

In der Abbildung 5.7 und 5.8 werden die Ergebnisse der Versuche mit neuem Aufbauen des Klassifikationsmodells erklärt. In diesen Versuchen wird das Klassifikationsmodell entsprechend der it=1, it=2, it=3, it=4 und it=5 neu aufgebaut, wobei es innerhalb der jeweiligen Iteration 1000 Dokumente gibt. Hier werden der Schwellwert des themenspezifischen Sprachmodells und der Schwellwert des Background Sprachmodells als 0.5 angenommen, wie sie in den Versuchen mit neuem Aufbauen der Sprachmodelle sind. Die "Standard" Kurven in den Abbildungen bezeichnen die Ergebnisse des Programms unter der Bedingung, dass die Sprachmodelle nach je 1000 Dokumenten neu aufgebaut werden und für die Schwellwerte von TLM und BLM 0.5 angenommen wird. Aus diesen Abbildungen kann man sehen, dass die Perplexitäten bei der jeweiligen Iterationsnummer nach jeder Iteration immer absteigt. Hierfür kann man auch sehen, dass je größer die Iterationsnummer "it" ist, desto niedriger die Perplexität bzw. desto besser die Qualität des themenspezifischen Sprachmodells ist. Jede ohne neues Aufbauen des Klassifikationsmo-

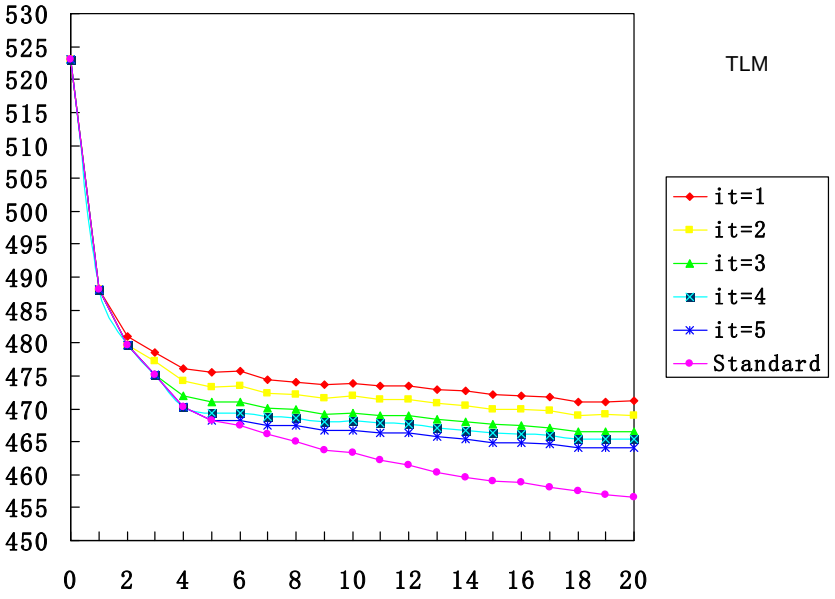


Abbildung 5.7.: Versuchsergebnisse nach dem neuen Aufbauen des Klassifikationsmodells für TLM

dells bekommende Perplexität ist immer niedriger als solche mit neuem Aufbauen, egal welchen Wert die Iterationsnummer angenommen hat. Die Versuchsergebnisse in der Abbildung 5.8 mit unterschiedlicher Iterationsnummer sind leider ganz ähnlich. Aber dazu kann man noch sehen, dass die "Standard" Kurve für BLM nicht mehr die beste ist, in der die Perplexitäten ohne neues Aufbauen des Klassifikationsmodells bewertet werden. Deswegen wurde hier ein neu interpoliertes Sprachmodell für jeden Versuch aufgebaut, das von in jedem Versuch letztlich bekommenden TLM und BLM integriert wurde. Danach wurden diese interpolierten Sprachmodelle mit den TLM Tuningdaten bewertet, um zu sehen, unter welcher Iterationsnummer sich das Programm am besten auswirken bzw.

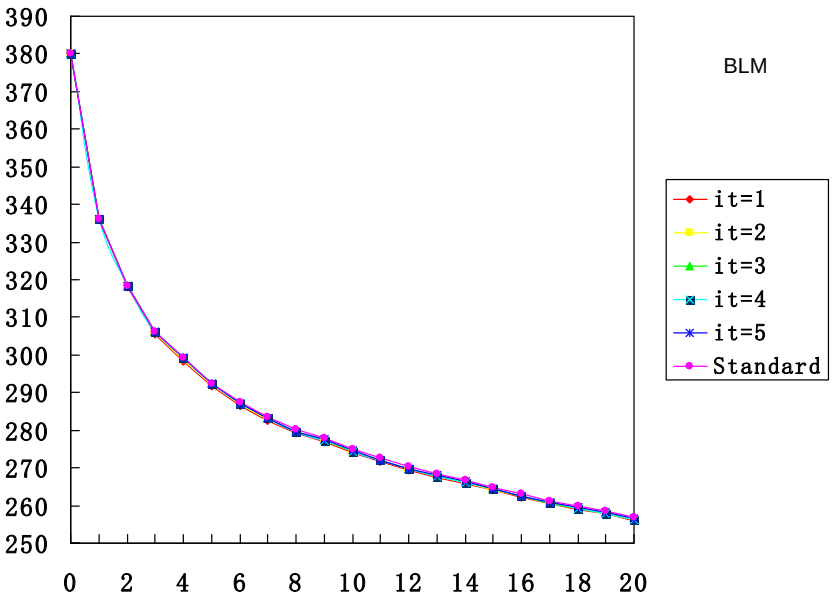


Abbildung 5.8.: Versuchsergebnisse nach dem neuen Aufbauen des Klassifikationsmodells für BLM

das beste themenspezifische Sprachmodell anbieten kann. Im Anhang A.7 zeigt es alle entsprechenden Versuchsergebnisse und aus dieser Daten kann man sehen, dass ohne neues Aufbauen des Klassifikationsmodells das Programm das beste themenspezifische Sprachmodell liefern kann. Damit kann man schließen, dass das Klassifikationsmodell im Verlauf des Programms immer gleich bleiben sollte, nachdem es mit den originalen Trainingsdateien aufgebaut worden ist, um die beste Leistung des Programms zu erreichen.

Wie im Unterkapitel 5.1 diskutiert, haben die neuen extrahierten Trainingsdaten schlechtere Qualität als solche originalen Trainingsdaten. Dies kann zu einem schlechteren Klassifikationsmodell führen. Zwar gibt es die Möglichkeit, dass ein Sprachmodell durch neues Aufbauen des Klassifikationsmodells so eine bessere Qualität erhalten kann, wie die Ergebnisse von BLM gezeigt haben, aber wegen der ganz kleinen Unterschiede zwischen den Ergebnissen von BLM kann man nicht genau sagen, ob diese erhaltenen Ergebnisse zufällig sind oder ob die Qualität der originalen Trainingsdaten eines Sprachmodells wirklich die Ergebnisse mit neuem Aufbauen des Klassifikationsmodells beeinflussen kann. Auch wenn die Qualität der originalen Trainingsdaten hierfür eine Rolle spielen kann, kann die Qualität des beliebig gegebenen themenspezifischen Sprachmodell ganz ohne Gewähr sein, so dass dieser Parameter Iterationsnummer, mit welchem das Klassifikationsmodell neu aufgebaut wird, ignoriert werden muss.

5.4. Evaluation der Methoden mit den schon bestimmten Parameter

Wie im Unterkapitel 4.4 schon erklärt, wird in dieser Evaluation die Leistung des ganzen Programms mit den in obigen Versuchen schon festgelegten Parameter eingeschätzt. Das Programm wird hier immer die originalen Sprachmodelle und das originale Klassifikationsmodell benutzen, um die 50000 Dokumente zu bewerten. Für solche Utterances in diesen eingelesenen Dokumenten werden durch den Vergleich mit dem Schwellwert von TLM 0.05 und dem Schwellwert von BLM 0.35 entschieden, ob sie als die Trainingsdaten vom themenspezifischen oder vom Background Sprachmodell extrahiert werden können oder ob

sie als ”nutzlose” Informationen ignoriert werden sollten.

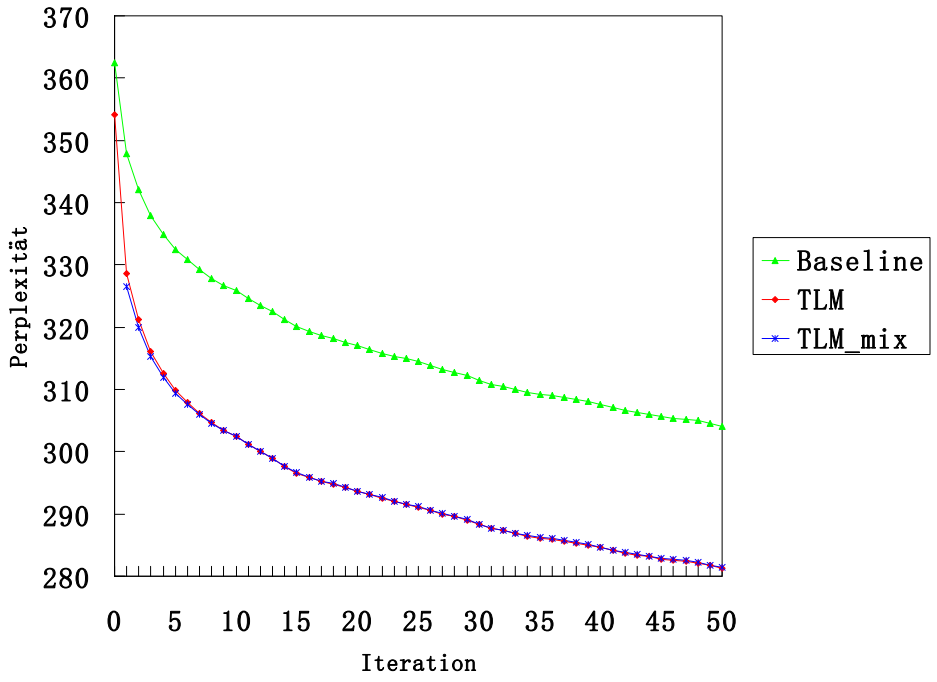


Abbildung 5.9.: Evaluierungsergebnisse mit schon festgelegten Parameter

Sprachmodell Typen	Perplexität am Anfang	Perplexität am Ende	Verbesserung
Baseline	362.422	304.006	16.12%
nur mit TLM	354.095	281.296	20.56%
mix TLM BLM	354.095	281.404	20.53%

Tabelle 5.1.: Vergleich der Sprachmodelle

In der Abbildung 5.9 wird die Evaluation erklärt. Darin bezeichnet ” Baseline” ein referenziertes Sprachmodell, welches am Anfang von den originalen Trainingsdaten vom TLM und BLM aufgebaut wurde und nachher immer nach jeweils 1000 Dokumenten von den originalen Trainingsdaten und den neu eingelesenen Daten neu aufgebaut wurde. Hier ist die Tabelle 5.1 basierend auf der Tabelle der detaillierten Versuchsergebnisse A.9 erzeugt worden. Aus der Abbildung und der Tabelle kann man sehen, dass vor dem Verlauf des Programms die bewertete Perplexität des Sprachmodells Baseline 362.422 erreicht. Während des Einlesens der Dokumente nimmt die nach jeweils einer Iteration (1000 Dokumente) neu bewertete Perplexität immer weiter ab. Die Tendenz der Kurve zu fallen ist am Anfang sehr rapid, aber beim Einlesen der Dokumente wird sie immer flacher. Letztlich kann die Perplexität des besten aufgebauten referenzierten Sprachmodells Baseline 304.006 erzielen. Davon kann man wissen, dass mit dieser einfachsten Methode, die alle vorhandenen Daten als Trainingsdaten für das Sprachmodell sieht, die bewerteten Perplexität um 16.12% abnehmen kann. Man kann auch sagen, dass das Sprachmodell sich durch den Verlauf um 16.12% verbessert.

Hier bezeichnet ” nur mit TLM” das Sprachmodell, welches von den originalen Trainingsdaten des themenspezifischen Sprachmodells(TLM) und den neu extrahierten themenbezogenen Daten aufgebaut wird. ” mix TLM BLM” beschreibt das Sprachmodell, welches von

der neu erhaltenen verbesserten Sprachmodellen Topic Sprachmodell(TLM) und Background Sprachmodell(BLM) interpoliert wird. Die berechneten Perplexitäten dieser zwei Sprachmodelle vor dem Verlauf des Programms haben beide 354.095 erreicht. Während des Verlaufs unterscheiden sich diese zwei Kurven ein bisschen. Die nach der jeweiligen Iteration neu berechnete Perplexität jedes Sprachmodells steigt nacheinander immer ab. In der Abbildung ist die Leistung des Sprachmodells " mix TLM BLM" zuerst besser als solche des Sprachmodells " nur mit TLM", aber nach einigen Iterationen ist es umgekehrt. Wie aus der Tabelle A.9 zu ersehen ist, ist die Perplexität des Sprachmodells " mix TLM BLM" vor der dreizehnten Iteration immer niedriger als solche des Sprachmodells " nur mit TLM". Ab der dreizehnten Iteration verhält sich das Sprachmodell " nur mit TLM" immer besser als das Sprachmodell " mix TLM BLM". Letztlich hat das Sprachmodell " nur mit TLM" eine Perplexität 281.296 und das Sprachmodell ' mix TLM BLM" eine Perplexität 281.404 erzielt, wobei das Erstere sich um 20.56% verbessert und das Letztere sich um 20.53% verbessert hat. Damit kann die Schlussfolgerung gezogen werden, dass das neu aufgebaute themenspezifische Sprachmodell ohne die Interpolation vom Background Sprachmodell eine bessere Leistung erhalten kann, wenn ausreichende einzulesenden Dokumente vorhanden sind.

Für den oben erwähnten Vergleich der Sprachmodelle " nur mit TLM" und " mix TLM BLM" kann die Ursache nach meiner Meinung im Folgenden analysiert werden. Um das Sprachmodell " mix TLM BLM" aufzubauen, muss das neu aufgebaute bzw. schon verbesserte themenspezifische Sprachmodell vom neu aufgebauten Background Sprachmodell interpoliert werden. Dafür muss zuerst das Gewicht für die Interpolation berechnet werden. Dieses Gewicht ändert sich nach jeder Iteration, nach der neue Trainingsdaten hinzugekommen sind und neue Sprachmodelle aufgebaut worden sind. Aus der Tabelle 5.2 ist zu ersehen, dass das Gewicht des themenspezifischen Sprachmodells immer höher wird und das Gewicht des Background Sprachmodells immer entsprechend niedriger wird. Das bedeutet, dass das Background Sprachmodell im interpolierten Sprachmodell " mix TLM BLM" immer geringere Rolle spielen kann. In der letzten Iteration ist das Gewicht(BLM) sogar kleiner als 0.01.

Gewicht (Sprachmodell)	1. Iteration	2. Iteration	3. Iteration	...
Gewicht(TLM)	0.966876	0.974048	0.977684	...
Gewicht(BLM)	0.033124	0.025952	0.022316	...
Gewicht (Sprachmodell)	12. Iteration	13. Iteration	...	50. Iteration
Gewicht(TLM)	0.988179	0.988489	...	0.991404
Gewicht(BLM)	0.011821	0.011511	...	0.008596

Tabelle 5.2.: Gewicht für Interpolation des Sprachmodells " mix TLM BLM"

Aus obiger Tabelle ergibt sich die Frage: Wieso ist ab der dreizehnten Iteration das interpolierte Sprachmodell schlechter als das themenspezifische Sprachmodell ohne die Interpolation mit dem Background Sprachmodell? Dies kann damit erklärt werden, dass wenn das Gewicht des Background Sprachmodells niedriger als ein bestimmter Wert ist, wird die vom Background Sprachmodell gelieferte Information nicht mehr nützlich sein. Diese Informationen sind so tolerant sogar themenunbezogen, dass sie zu einem immer schlechteren neu interpolierten Sprachmodell führen können. Deswegen sollte man nur das themenspezifische Sprachmodell nehmen, wenn ausreichend Dokumente vorhanden sind, weil sich nach einer bestimmten Anzahl an Iterationen das Background Sprachmodell nicht mehr positiv auswirken kann.

In der Abbildung 5.9 kann man auch aus der Tendenz der Kurven sehen, dass im Verlauf

des Programms die Kurven immer flacher werden. Dies kann zu einem Problem führen, dass zwar durch diese Methode die Qualität eines themenspezifischen Sprachmodells verbessert werden kann, aber auch, dass es zum Konvergieren der Perplexität führen kann. In diesem Fall ist es schwer sehr schnell ein viel verbessertes Sprachmodell zu erhalten, insbesondere weil nach einigen Iterationen schon ein relativ gutes Sprachmodell aufgebaut worden ist.

6. Zusammenfassung

Themenspezifische Sprachmodelle ergeben immer gute Leistungen im Bereich der Spracherkennung und auch in den anderen Anwendungsbereichen. Aber für manche Themenbereiche gibt es nur eine eingeschränkte Menge an Trainingsdaten. Mit dieser Menge kann ein themenspezifisches Sprachmodell mit einer schlechten oder mäßigen Qualität aufgebaut werden. Im diesem Fall muss eine Voradaption der Trainingsdaten eines Themenbereiches durchgeführt werden. Da viele unklassifizierte Dokumente vorhanden sind, wird eine Methode nach [SGN05] entworfen. Für die eingelesenen Dokumente werden zwei Formeln 6.1 verwendet, um zu entscheiden, welche Utterances in diesen Dokumenten zu dem Thema passen, welche zu dem Background und welche nicht zu beiden passen. In diesen Formeln sind die von einem Sprachmodell bewertete Perplexität eines Utterances und die durch Textklassifikation erhaltene Wahrscheinlichkeit eines Dokumentes nötig. Hierfür können die Werkzeuge **SRILM** und **Rainbow** benutzt werden, um die Daten in den folgenden Formeln vorzubereiten. Dabei wird für die Utterances in den unklassifizierten Dokumenten durch die Formel **Score(T)** entschieden, ob sie zu dem Thema passen und als neue Trainingsdaten des themenspezifischen Sprachmodells extrahiert werden können. Analog wird für sie durch die Formel **Score(B)** entschieden, ob sie zu dem Background passen, so dass sie als neue Trainingsdaten des Background Sprachmodells extrahiert werden können. Die Utterances, deren Bewertungsnoten Score(T) oder Score(B) den entsprechenden Schwellwert überschreiten können, werden als für das Thema oder für den Background relevant angesehen. Diese extrahierten Utterances können dabei helfen, das themenspezifischen Sprachmodell und das Background Sprachmodell mit besserer Qualität neu aufzubauen.

$$
\begin{aligned}
Score(T) &= \frac{\frac{1}{P(utt|T)} * DW(T)}{\frac{1}{P(utt|T)} * DW(T) + \frac{1}{P(utt|B)} * DW(B)} \\
Score(B) &= \frac{\frac{1}{P(utt|B)} * DW(B)}{\frac{1}{P(utt|T)} * DW(T) + \frac{1}{P(utt|B)} * DW(B)}
\end{aligned}
\tag{6.1}
$$

Bis jetzt wird das grundlegende Prinzip schon erklärt. Für diesen Entwurf sind noch ein paar Varianten benötigt, um zu entscheiden, welche Varianten nützlich sind, welche Werte sie annehmen sollten und welche Varianten sinnlos sind, so dass die entworfene Methode die beste Leistung liefern kann. Hierfür wurden viele Versuche durchgeführt, um die Varianten festzulegen. Dadurch wird die Schlussfolgerung gezogen, dass die Variante des neuen Aufbauens der Sprachmodelle und die Variante des neuen Aufbauens des Klassifikationsmodells vergeblich sind und die Variante Schwellwert dabei helfen kann, ein besseres Sprachmodell zu erhalten.

In der Evaluation mit den schon bestimmten Varianten werden drei Sprachmodelle verglichen. Darin werden 50000 Dokumente eingelesen und jeweils 1000 Dokumente für eine Iteration benutzt. Es gibt insgesamt 50 Iterationen. Nach jeder Iteration werden die Sprachmodelle mit den neu hinzugefügten Trainingsdaten aktualisiert und jedes neu aufgebaute Sprachmodell wird dieselbe Testdatei verwendet, um die Perplexität zu berechnen. Ein Sprachmodell von den drei Sprachmodellen ist "Baseline", welches von allen vorhandenen Daten,z.B. von den originalen Trainingsdaten des themenspezifischen Sprachmodells und des Background Sprachmodells und auch von allen eingelesenen Daten, aufgebaut wird. Es wird im Verlauf des Einlesens der Dokumente nach jeder Iteration neu aufgebaut, wobei die Trainingsdaten die originalen Trainingsdaten und auch alle neu hereinkommenden Daten umfassen. Diese "Baseline" wird als ein referenziertes Sprachmodell gesehen, welches sehr einfach ist und auch zur Verbesserung des themenspezifischen Sprachmodells führen kann. Mit ihm können die mit den entworfenen Methoden aufgebauten Sprachmodelle verglichen werden, um zu sehen, ob diese Sprachmodelle mehr als das Sprachmodell "Baseline" verbessert werden können. Diese zwei Sprachmodelle sind " nur mit TLM" und " mix TLM BLM". Das Erstere ist ein themenspezifisches Sprachmodell, das immer mit den neuen extrahierten themenbezogenen Trainingsdaten aufgebaut wird, ohne Berücksichtigung des Background Sprachmodells. Das Letztere ist ein Sprachmodell, das vom themenspezifischen Sprachmodell und vom Background Sprachmodell hergeleitet wird, die nach jeder Iteration neu aufgebaut worden sind und sich dadurch verbessert haben. Am Anfang ist das Sprachmodell " nur mit TLM" nicht so gut wie das Sprachmodell " mix TLM BLM", aber nach einigen Iterationen ist das Sprachmodell " nur mit TLM" immer besser als das Sprachmodell " mix TLM BLM". Deswegen kann man sehen, dass für ausreichend einzulesende Dokumente das Sprachmodell " nur mit TLM" am besten funktionieren kann. In der Evaluation dieser Arbeit kann die Qualität des erhaltenen Sprachmodells sich am besten um **20.56%** verbessern, im Vergleich mit dem Verbesserungsgrad **16.12%**, die von der einfachsten Methode erzielt wird. Damit kann die Schlussfolgerung gezogen werden, dass die Methode dieser Arbeit eine guten Leistung erzielt hat.

In der Zukunft kann man unterschiedliche Ngrammsgrade annehmen, um zu sehen, ob die Qualität der Sprachmodelle noch verbessert werden kann, da die Sprachmodelle alle als Trigram Sprachmodelle aufgebaut wurden. Es kann sein, dass die Sprachmodelle mit anderen Ngramsgraden bessere Qualität erzielen können. Aber es gibt auch ein Problem. Mit der Beobachtung der immer kleiner werdenden Änderungen der Perplexität ist es möglich, dass es immer länger dauern und schwieriger werden kann, wenn man eine immer niedrigere Perplexität bzw. immer bessere Sprachmodelle bekommen möchte. Im diesen Fall kann man eine andere Methode ausprobieren, so dass nur solche Utterances, deren Perplexitäten weniger als ein bestimmter Schwellwert sind, zum Bewertungsverfahren mit den Formeln 6.1 weitergeleitet werden. Unter dieser Bedingung können die extrahierten Utterances eine bessere Qualität erhalten, aber es werden mehr unklassifizierte Dokumente benötigt, um ein zufrieden stellendes Ergebnis zu erreichen, da weniger Utterances diese Vorbedingung erfüllen können. Diese Methode kann vermutlich das obige Problem, dass es immer länger dauern und schwieriger wird eine immer niedrigere Perplexität zu erhalten, zum Teil erleichtern.

Literaturverzeichnis

[AG05] Alexandre Allauzen und Jean Luc Gauvain: *Robust topic inference for latent semantic language model adaptation.* INTERSPEECH, 2005.

[Bel04] Jerome R. Bellegarda: *Statistical language model adaptation: review and perspectives.* Speech Communication, 42:93–108, 2004.

[Che98] Stanley F. Chen: *An Empirical Study of Smoothing Techniques for Language Modeling.* Technischer Bericht, 1998.

[GGLL02] Jianfeng Gao, Joshua Goodman, Mingjing Li und Kai Fu Lee: *Toward a unified approach to statistical language modeling for Chinese.* 1:3–33, March 2002, ISSN 1530-0226. `http://doi.acm.org/10.1145/595576.595578`.

[Hei07] A. Lin shan Lee Heidel: *Robust topic inference for latent semantic language model adaptation.* Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop, Seiten 177 – 182, 2007.

[Jef48] H. Jeffreys: *Theory of probability.* Clarendon Press, Oxford, second edition, 1948.

[Joh32] W.E. Johnson: *Probability: deductive and inductive problems.* Mind, 41:421–423, 1932.

[Kla00] Dietrich Klakow: *Selecting articles from the language model training corpus.* Band 3, Seiten 1695–1698, Istanbul, Turkey, 2000. ICASSP2000.

[Koe05] Philipp Koehn: *Europarl: A Parallel Corpus for Statistical Machine Translation.* AAMT, 2005. `http://mt-archive.info/MTS-2005-Koehn.pdf`.

[Lid20] George James Lidstone: *Note on the general case of the bayes-laplace formula for inductive or a posteriori probabilities.* Transactions of the Faculty of Actuaries, 8:182–192, 1920.

[McC] Andrew Kachites McCallum: *Rainbow.* `http://www.cs.cmu.edu/~mccallum/bow/rainbow/`.

[ML10] Robert C. Moore und William Lewis: *Intelligent Selection of Language Model Training Data.* In: *ACL (Short Papers)'10*, Seiten 220–224, 2010.

[SGG] Ruhi Sarikaya, Agustin Gravano und Yuqing Gao: *Rapid Language Model Development Using External Resources for New Spoken Dialog Domains.* In: *in Proc. ICASSP, 2005*, Seiten 573–576.

[SGN05] Abhinav Sethy, Panayiotis G. Georgiou und Shrikanth S. Narayanan: *Building topic specific language models from webdata using competitive models.* In: *Proceedings of InterSpeech*, Seiten 1293–1296, Lisbon, Portugal, Oktober 2005.

[SR02] Fabrizio Sebastiani und Consiglio Nazionale Delle Ricerche: *Machine Learning in Automated Text Categorization.* ACM Computing Surveys, 34:1–47, 2002.

[SRI] *SRILM - The SRI Language Modeling Toolkit*. http://www-speech.sri.com/projects/srilm/.

[Stü09] Sebastian Stüker: *Grundlagen der Automatischen Spracherkennung, Sprachmodellierung I*. Seiten 2,6,15, 2009. http://isl.ira.uka.de/fileadmin/cmu-uka-shared-files/St%FCker/2009-12-09.pdf.

[Sto02] Andreas Stolcke: *SRILM – AN EXTENSIBLE LANGUAGE MODELING TOOLKIT*. Speech Technology and Research Laboratory, 2002. http://www.speech.sri.com/.

[wika] wikipedia: *Language model*. http://en.wikipedia.org/wiki/Language_model.

[wikb] wikipedia: *Spracherkennung*. http://de.wikipedia.org/wiki/Spracherkennung.

[wikc] wikipedia: *Word Error Rate*. http://en.wikipedia.org/wiki/Word_error_rate.

Anhang

A. Tabellen der Versuchsergebnisse

TLM: PPL Iteration doc

20 000 doc	0	1k	2k	3k	4k	5k	6k	7k	8k	9k	10k	
I=100 doc/it	522.99	500.488	493.118	489.582	485.462	483.82	483.17	481.434	480.287	478.818	478.414	
I=500 doc/it	522.99	491.665	483.269	479.788	476.098	474.352	473.399	471.807	470.696	469.618	469.382	
I=1000 doc/it	522.99	488.133	479.595	475.201	470.259	468.184	467.45	466.107	465.039	463.649	463.325	
I=5000 doc/it	522.99					449.935					441.776	
I=10000 doc/it	522.99										432.065	
I=20000 doc/it	522.99											
20 000 doc	0	11k	12k	13k	14k	15k	16k	17k	18k	19k	20k	TLM BLM Mix
I=100 doc/it	522.99	477.272	476.136	475.165	473.998	473.201	472.963	471.965	471.067	470.479	470.076	393.761
I=500 doc/it	522.99	468.397	467.453	466.347	465.374	464.867	464.855	463.887	463.359	462.761	462.359	393.428
I=1000 doc/it	522.99	462.146	461.364	460.329	459.499	458.971	458.919	458.057	457.529	456.948	456.586	393.092
I=5000 doc/it	522.99					436.401					433.459	390.653
I=10000 doc/it	522.99										422.43	388.629
I=20000 doc/it	522.99										410.352	386.039

Abbildung A.1.: Versuchsergebnisse vom Parameter Iterationsnummer des neuen Aufbauen der Sprachmodelle

BLM: PPL Iteration doc

20 000 doc	0	1k	2k	3k	4k	5k	6k	7k	8k	9k	10k
I=100 doc/it	380.209	335.896	318.291	306.321	299.063	292.276	287.195	283.354	280.025	277.797	274.9
I=500 doc/it	380.209	335.793	318.229	306.12	299.032	292.219	287.298	283.408	280.049	277.884	275.026
I=1000 doc/it	380.209	336	318.511	306.316	299.264	292.488	287.379	283.513	280.095	277.933	275.11
I=5000 doc/it	380.209					293.393					275.996
I=10000 doc/it	380.209										276.614
I=20000 doc/it	380.209										
20 000 doc	0	11k	12k	13k	14k	15k	16k	17k	18k	19k	20k
I=100 doc/it	380.209	272.39	269.996	268.198	266.58	264.785	262.892	261.078	259.6	258.389	256.68
I=500 doc/it	380.209	272.501	270.157	268.322	266.72	264.757	262.88	261.13	259.628	258.406	256.683
I=1000 doc/it	380.209	272.615	270.281	268.477	266.859	264.915	263.058	261.27	259.768	258.557	256.811
I=5000 doc/it	380.209					265.645					257.535
I=10000 doc/it	380.209										258.273
I=20000 doc/it	380.209										258.677

Abbildung A.2.: Versuchsergebnisse vom Parameter Iterationsnummer des neuen Aufbauen der Sprachmodelle

TLM: PPL Threshold Iteration

20 000 doc	0	1	2	3	4	5	6	7	8	9	10
threshold=0.1	522.99	435.418	417.298	405.616	397.538	392.277	387.894	383.637	379.744	376.161	373.811
threshold=0.2	522.99	438.842	418.802	406.296	398.037	392.652	388.192	383.881	379.94	376.343	373.935
threshold=0.3	522.99	445.759	426.362	413.016	404.758	398.703	393.762	389.352	385.046	381.167	378.348
threshold=0.4	522.99	458.206	443.997	433.583	427.242	423.214	419.172	415.561	413.558	410.204	407.495
threshold=0.5	522.99	473.112	463.019	459.046	456.537	454.365	452.107	450.144	449.508	447.976	447.099
threshold=0.6	522.99	491.665	486.329	485.541	483.888	481.979	480.494	478.87	478.256	476.822	476.246
threshold=0.7	522.99	505.792	500.292	500.6	499.323	497.488	496.648	495.921	495.022	493.589	493.274
threshold=0.8	522.99	517.991	515.677	514.038	512.726	511.994	510.859	510.82	509.922	509.373	508.943
threshold=0.9	522.99	522.488	522.179	522.101	522.121	522.117	522.071	522.046	522.003	521.966	521.966

Abbildung A.3.: Versuchsergebnisse vom Parameter TLM Threshold

BLM: PPL Threshold Iteration

20 000 doc	0	1	2	3	4	5	6	7	8	9	10
threshold=0.1	380.209	317.219	298.376	287.087	280.051	275.104	270.444	267.049	263.356	260.211	257.328
threshold=0.2	380.209	317.239	298.399	287.101	280.02	275.094	270.435	267.034	263.341	260.262	257.373
threshold=0.3	380.209	317.529	298.613	287.342	280.431	275.371	270.599	267.119	263.43	260.351	257.389
threshold=0.4	380.209	318.621	299.022	287.249	279.998	274.954	270.229	266.734	263.103	259.835	256.923
threshold=0.5	380.209	319.356	299.573	287.84	280.432	275.389	270.542	267.154	263.362	260.086	257.078
threshold=0.6	380.209	322.624	301.197	289.184	281.626	276.504	271.466	267.842	263.985	260.601	257.585
threshold=0.7	380.209	327.965	305.255	292.009	283.638	278.235	272.921	269.072	265.036	261.746	258.708
threshold=0.8	380.209	343.856	316.108	301.573	291.86	285.393	279.072	274.414	270.234	266.906	263.355
threshold=0.9	380.209	367.796	352.96	337.451	323.425	313.638	305.133	297.674	292.08	287.494	282.211

Abbildung A.4.: Versuchsergebnisse vom Parameter BLM Threshold

TLM: PPL Threshold Iteration (detailliert)

20 000 doc	0	1	2	3	4	5	6	7	8	9	10
threshold=0.05	522.99	435.063	417.17	405.546	397.469	392.212	387.881	383.64	379.751	376.162	373.802
threshold=0.1	522.99	435.418	417.298	405.616	397.538	392.277	387.894	383.637	379.744	376.161	373.811
threshold=0.15	522.99	436.706	417.867	405.802	397.689	392.32	387.935	383.701	379.786	376.182	373.825
threshold=0.2	522.99	438.842	418.802	406.296	398.037	392.652	388.192	383.881	379.94	376.343	373.935

Abbildung A.5.: Versuchsergebnisse vom Parameter TLM Threshold (detailliert)

BLM: PPL Threshold Iteration (detailliert)

20 000 doc	0	1	2	3	4	5	6	7	8	9	10
threshold=0.3	380.209	317.529	298.613	287.342	280.431	275.371	270.599	267.119	263.43	260.351	257.389
threshold=0.35	380.209	318.222	299.022	287.503	280.28	275.058	270.206	266.743	263.009	259.839	256.921
threshold=0.4	380.209	318.621	299.022	287.249	279.998	274.954	270.229	266.734	263.103	259.835	256.923
threshold=0.45	380.209	318.401	299.142	287.343	280.161	275.179	270.334	266.777	263.114	259.854	256.943

Abbildung A.6.: Versuchsergebnisse vom Parameter BLM Threshold (detailliert)

TLM mit neu aufgebautem Klassifikationsmodell

20 000 doc	0	1	2	3	4	5	6	7	8	9	10
it=1	522.99	488.133	481.008	478.485	476.189	475.468	475.685	474.443	474.119	473.662	473.938
it=2	522.99	488.133	479.595	477.199	474.166	473.35	473.466	472.441	472.173	471.674	472.008
it=3	522.99	488.133	479.595	475.201	472.033	471.076	471.038	470.021	469.864	469.103	469.409
it=4	522.99	488.133	479.595	475.201	470.259	469.305	469.418	468.775	468.597	467.954	468.263
it=5	522.99	488.133	479.595	475.201	470.259	468.184	468.219	467.532	467.436	466.659	466.803
Standard	522.99	488.133	479.595	475.201	470.259	468.184	467.45	466.107	465.039	463.649	463.325
20 000 doc	0	11	12	13	14	15	16	17	18	19	20	TLM BLM Mix
it=1	522.99	473.433	473.418	472.888	472.644	472.121	472.037	471.773	471.105	471.105	471.141	394.806
it=2	522.99	471.473	471.428	470.779	470.456	469.946	469.88	469.63	468.982	469.112	469.027	394.661
it=3	522.99	468.953	468.952	468.346	468.046	467.569	467.471	467.146	466.48	466.598	466.553	394.541
it=4	522.99	467.771	467.573	467.009	466.767	466.255	466.237	466.007	465.322	465.393	465.35	394.447
it=5	522.99	466.352	466.333	465.71	465.417	464.905	464.861	464.642	464.016	464.084	464.046	394.278
Standard	522.99	462.146	461.364	460.329	459.499	458.971	458.919	458.057	457.529	456.948	456.586	393.092

Abbildung A.7.: Versuchsergebnisse vom Parameter Iterationsnummer des neuen Aufbauen des Klassifikationsmodells

BLM mit neu aufgebautem Klassifikationsmodell

20 000 doc	0	1	2	3	4	5	6	7	8	9	10
it=1	380.209	336	318.158	305.536	298.356	291.592	286.429	282.641	279.274	277.023	274.069
it=2	380.209	336	318.511	305.819	298.787	292.009	286.837	283.001	279.604	277.332	274.37
it=3	380.209	336	318.511	306.316	299.134	292.34	287.055	283.164	279.735	277.432	274.473
it=4	380.209	336	318.511	306.316	299.264	292.397	287.109	283.175	279.705	277.404	274.477
it=5	380.209	336	318.511	306.316	299.264	292.488	287.167	283.233	279.736	277.456	274.572
Standard	380.209	336	318.511	306.316	299.264	292.488	287.379	283.513	280.095	277.933	275.11
20 000 doc	0	11	12	13	14	15	16	17	18	19	20
it=1	380.209	271.656	269.33	267.518	265.933	264.065	262.177	260.392	258.976	257.793	256.074
it=2	380.209	271.918	269.572	267.752	266.14	264.256	262.371	260.579	259.141	257.964	256.248
it=3	380.209	272.016	269.654	267.826	266.241	264.35	262.44	260.641	259.215	258.028	256.323
it=4	380.209	271.976	269.633	267.858	266.241	264.398	262.514	260.716	259.287	258.049	256.354
it=5	380.209	272.094	269.727	267.978	266.395	264.55	262.645	260.84	259.395	258.135	256.437
Standard	380.209	272.615	270.281	268.477	266.859	264.915	263.058	261.27	259.768	258.557	256.811

Abbildung A.8.: Versuchsergebnisse vom Parameter Iterationsnummer des neuen Aufbauen des Klassifikationsmodells

Evaluation mit Threshhold TLM=0.05 BLM=0.35, ohne neuem Aufbauen des Sprachmodells und des Klassifikationsmodell

50 000 doc	0	1	2	3	4	5	6	7	8	9	10
Baseline	362.422	347.828	342.111	337.861	334.892	332.496	330.83	329.196	327.872	326.647	325.865
nur mit TLM	354.095	328.589	321.194	316.121	312.503	309.8	307.864	306.118	304.667	303.411	302.506
mix TLM BLM		326.539	319.954	315.259	311.983	309.431	307.655	305.978	304.581	303.346	302.463
		11	12	13	14	15	16	17	18	19	20
Baseline	362.422	324.535	323.424	322.47	321.168	320.068	319.358	318.646	318.2	317.585	317.001
nur mit TLM	354.095	301.169	300.004	298.993	297.606	296.59	295.907	295.23	294.803	294.232	293.643
mix TLM BLM		301.145	299.997	298.997	297.627	296.617	295.939	295.271	294.851	294.281	293.697
		21	22	23	24	25	26	27	28	29	30
Baseline	362.422	316.429	315.822	315.306	314.911	314.438	313.864	313.235	312.79	312.223	311.517
nur mit TLM	354.095	293.096	292.539	292.021	291.527	291.113	290.57	289.988	289.554	289.015	288.297
mix TLM BLM		293.158	292.6	292.087	291.595	291.18	290.643	290.061	289.631	289.095	288.378
		31	32	33	34	35	36	37	38	39	40
Baseline	362.422	310.82	310.486	309.956	309.588	309.218	309.095	308.741	308.351	308.044	307.584
nur mit TLM	354.095	287.651	287.342	286.82	286.427	286.097	285.974	285.647	285.31	285.03	284.63
mix TLM BLM		287.736	287.425	286.912	286.523	286.193	286.073	285.743	285.411	285.128	284.732
		41	42	43	44	45	46	47	48	49	50
Baseline	362.422	307.047	306.623	306.328	306.031	305.649	305.417	305.221	304.982	304.482	304.006
nur mit TLM	354.095	284.131	283.749	283.426	283.133	282.775	282.598	282.414	282.163	281.711	281.296
mix TLM BLM		284.234	283.851	283.532	283.241	282.885	282.704	282.521	282.271	281.818	281.404

Abbildung A.9.: Evaluierungsergebnisse mit festgelegten Parameter