Universität Karlsruhe (TH)

Diplomarbeit

Automatische Fehlererkennung und Duplikateliminierung auf interaktiv gelernten Wissensbasen

Studienarbeiter:	Philipp Große
Betreuer:	Prof. Dr. A. Waibel
Betreuer:	Dipl.-Inform. H. Holzapfel
Tag der Abgabe:	31.05.2009

Wintersemester 2008/2009

Fakultät für Informatik
Institut für Theoretische Informatik (ITI)

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Diplomarbeit selbständig verfasst habe und nur die in der Arbeit angegebenen Hilfsmittel und Literaturstellen verwendet habe.

Karlsruhe, den 31. Mai 2009

Philipp Grafe

Vorname Nachname

Inhaltsverzeichnis

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

2 Terminologie und Grundlagen 5
- 2.1 IslEnquirer . . . 5
- 2.2 Natürlichsprachige Dialogsysteme . . . 7
  - 2.2.1 Automatische Spracherkennung . . . 8
  - 2.2.2 Verstehen natürlicher Sprache . . . 9
  - 2.2.3 Sprachsynthese . . . 9
- 2.3 Die Identität des Gesprächspartners . . . 10
- 2.4 Wissensbasis . . . 12
  - 2.4.1 Fehlertypen . . . 12
- 2.5 Vektordarstellung von Bilddaten . . . 15
  - 2.5.1 Diskrete Kosinustransformation . . . 16
  - 2.5.2 Zig-Zag-Scan . . . 17
- 2.6 Logistische Regression . . . 18
- 2.7 Distanzmaße . . . 19

3 Verwandte Arbeiten 21
- 3.1 Clustering . . . 21
  - 3.1.1 Partitionierendes Clustering . . . 22
    - 3.1.1.1 k-Means . . . 22
  - 3.1.2 Hierarchisches Clustering . . . 23
    - 3.1.2.1 BIRCH . . . 24
    - 3.1.2.2 ROCK . . . 26
    - 3.1.2.3 Chameleon . . . 27
  - 3.1.3 Dichtebasiertes Clustering . . . 28
  - 3.1.4 Modellbasiertes Clustering . . . 28
  - 3.1.5 Semi-supervised Clustering . . . 29
- 3.2 Active Learning . . . 30
- 3.3 Duplikateliminierung auf Datenbanken . . . 31

3.3.1 Die Sorted-Neighborhood Methode 33
3.3.2 Die Active Learning Methode 34

4 Erkennung und Auflösung von Wissensbasisfehlern 35
4.1 Problembeschreibung und Datenlage 35
4.2 Fehlererkennung durch Clustering auf Bilddaten 37
4.2.1 Cluster-Distanzen 38
4.2.2 Iteratives Clustering 41
4.2.3 Abbruchkriterium & Adaptive Distanz 43
4.3 Auflösung der fehlerhaften Session-Label 45
4.3.1 Clusterbasiertes Mapping 48
4.3.2 Korrektur-Dialoge 49
4.3.2.1 Sortierung der Label nach Priorität 51
4.3.2.2 Existenz-Dialog 53
4.3.3 Dialogbasiertes Mapping 53

5 Experimente und Ergebnisse 57
5.1 Evaluationsmetriken 57
5.2 Beschreibung der Daten 59
5.3 Distanzmaße & Konfidenz 62
5.3.1 Training der Konfidenz 62
5.3.2 Auswahl der Konfidenz-Merkmale 63
5.3.3 Vergleich der Distanzmaße 66
5.4 Mapping & Existenz-Dialoge 71
5.4.1 Mapping-Verfahren 71
5.4.2 Qualität und Quantität der gestellten Fragen 77

6 Diskussion & Ausblick 79
6.1 Zusammenfassung und Diskussion der Ergebnisse 79
6.2 Ausblick 80

Index 85

Abbildungsverzeichnis

1.1 „Verschmutzung“ der Wissensbasis im zeitlichen Verlauf . . 2

2.1 Ein Foto vom IslEnquirer 6
2.2 Aufbau eines Dialogsystems 8
2.3 Vier unterscheidbare Fehlertypen auf der Wissensbasis . . 13
2.4 Basis der Diskreten Kosinustransformation 17
2.5 Sortierung durch Zig-Zag-Scan 17

3.1 Ablauf des k-Means-Algorithmus 22
3.2 Hierarchisches Clustering 24
3.3 Ablauf des Chameleon-Algorithmus 27

4.1 Cluster-Varianzen 39
4.2 Centroidbasiertes iteratives agglomeratives Clustering . . . 42
4.3 Cluster und Session-Label in Groundtruth Darstellung . . 46
4.4 Cluster und Session-Label ohne Wissen um Groundtruth . 47
4.5 Aufbau der Korrektur-Dialoge 50
4.6 Cluster und Session-Label nach Existenz-Dialog 54

5.1 „Verunreinigung“ des Set1 im zeitlichen Verlauf 61
5.2 „Verunreinigung“ des Set2 im zeitlichen Verlauf 61
5.3 Konfidenz-Merkmale im ROC Graph 64
5.4 Cluster-Fusion bei Euklid-Distanz 67
5.5 Cluster-Fusion bei Hybrid-Distanz 69
5.6 Cluster-Fusion bei durchschnitlicher Editier-Distanz 69
5.7 Cluster-Fusion bei durchschnitlicher inter-Cluster-Distanz . 69
5.8 Distanzmaß Vergleich 71
5.9 Entry Error Rate bei Set1 73
5.10 F-Measure Error Rate bei Set1 73
5.11 Session Error Rate bei Set1 73
5.12 Entry Error Rate bei Set2 76
5.13 F-Measure Error Rate bei Set2 76
5.14 Session Error Rate bei Set2 76

5.15 Benefit-Rate auf Set1 78

Tabellenverzeichnis

2.1 Beispiel für Namenlern-Dialog des IslEnquirer 11

4.1 Beispiel für Existenz-Dialog des IslEnquirer 53

5.1 Beschreibung der Sets 59
5.2 Logit-Koeffizienten für Konfidenzen bei centroidbasierter Euklid-Distanz 64
5.3 Logit-Koeffizienten für Konfidenzen bei Hybrid-Distanz . . 65
5.4 Logit-Koeffizienten für Konfidenzen bei Editier-Distanz . . 65
5.5 Logit-Koeffizienten für Konfidenzen bei durchschnittlicher inter-Cluster-Distanz 65
5.6 Label-Mapping Ergebnisse für Set1 75
5.7 Label-Mapping Ergebnisse für Set2 77
5.8 Benefit-Rate für Set1 und Set2 78

1 Einleitung

1.1 Motivation

Ein langjähriger Traum vieler Wissenschaftler ist es einen Roboter nach dem Vorbild des Menschen zu erschaffen. Neben dem menschlich motivierten Äußeren ist bei dieser Art von Roboter auch eine möglichst menschliche Verhaltensweise gewünscht. Der Roboter soll laufen, Gestik zeigen und sogar Gefühle empfinden können, wie man dies vom menschlichen Vorbild gewohnt sind. Ein äußerst wichtiger Aspekt in diesem Zusammenhang ist die Kommunikation und Interaktion mit dem Menschen. Ist die Maschine in der Lage, Menschen zu erkennen und zu identifizieren, so ermöglicht dies dem Roboter auf natürliche Art und Weise mit Personen zu kommunizieren.

Der Schwerpunkt der meisten, wenn nicht aller, Arbeiten im Umfeld humanoider Roboter ist es dem Roboter neue Fähigkeiten zur Verfügung zu stellen bzw. Fertigkeiten erlernen zu lassen. So existieren inzwischen beispielsweise Systeme (unter anderem der dieser Arbeit zugrundeliegende IslEnquirer), die dank Spracherkennung und Dialogsystem in einem natürlichen Dialog mit dem Benutzer interagieren und dank FaceID und VoiceID das Gesicht oder die Stimme eines Menschen wiedererkennen können. Basierend auf dieser Fähigkeit, ist es diesen Systemen möglich eigenständig neue Personen kennenzulernen und sie bei einem erneuten Treffen wiederzuerkennen. Ein solches System lernt im Laufe der Zeit immer mehr und die Wissensbasis des Roboters wächst während sich der Roboter eigenständig mit neuen Personen unterhält. Immer mehr Personen, Gesichter und Namen werden so im Speicher des Roboters gehalten.

Neben der Fähigkeit zu lernen, beherrscht das menschliche Vorbild jedoch noch eine weitere wertvolle Fähigkeit. Eine Fähigkeit die im Bereich der humanoiden Roboter bisher weitestgehend vernachlässigt wurde. Die Fähigkeit zu vergessen.

Aber wieso sollte das Vergessen eine wertvolle Fähigkeit sein? Wer hat sich denn noch nie darüber geärgert etwas wichtiges vergessen zu haben? Und wieso sollte das eine Fähigkeit sein, die ein Robotor erst erlernen

muss? Ist das Überschreiben der Festplatte nicht etwa das maschinelle Gegenstück zu dem menschlichen Vergessen?

Wenn wir im Zusammenhang dieser hier vorliegenden Arbeit von „vergessen" sprechen, dann ist nicht etwa das „blinde" Löschen von Daten gemeint, sondern vielmehr die Fähigkeit Wichtiges von Unwichtigem zu trennen und Letzteres im Zweifelsfall auszublenden.

Und das ist, so wird niemand bestreiten wollen, eine extrem wertvolle Fähigkeit. Nicht nur ermöglicht eine solche Fähigkeit erst sinnvolle Entscheidungen zu treffen, sondern sie hilft auch den Überblick über eine Vielzahl heterogener Informationsquellen zu wahren. Und auch bei einem autonom lernenden Roboter, der mit jeder Interaktion mit einem Menschen dazulernt, ist diese Fähigkeit extrem wertvoll, möchte man dem Roboter das Lernen über einen längeren Zeitraum ermöglichen.

Um das Gesagte anhand des am Institut entwickelten IslEnquirers zu demonstrieren, zeigt Abbildung 1.1 die zeitliche Entwicklung eines Ausschnitts der Wissensbasis des IslEnquirers. Der IslEnquirer hat, wie in Kapitel 2.1 ausführlicher erläutert wird, unter anderem das Ziel in einem natürlichen Dialog Personen kennenzulernen, wobei er die gelernten Namen als Label der Sessions in seiner Wissensbasis aufnimmt. Ein abgeschlossener Dialog mit einer Person entspricht dabei einer (Aufnahme-)Session.

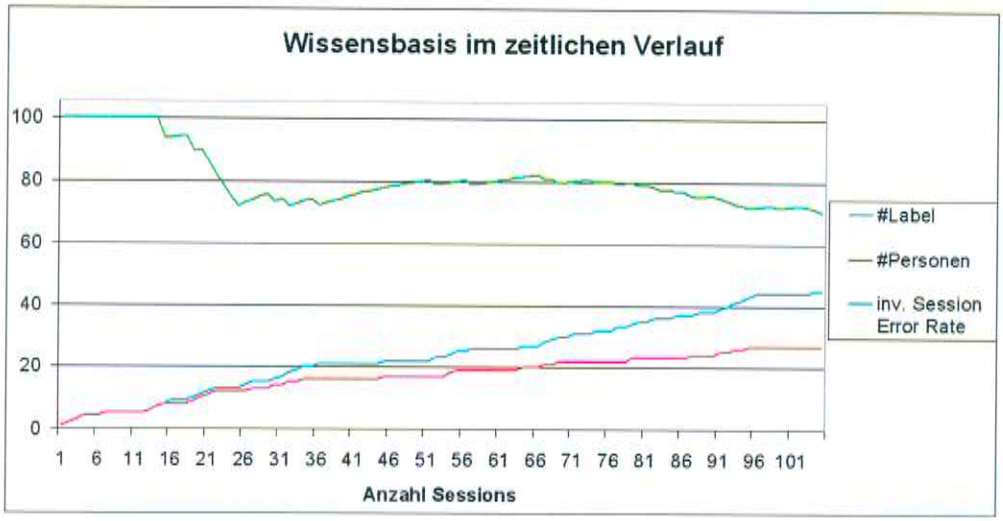

Abbildung 1.1: „Verschmutzung" der Wissensbasis im zeitlichen Verlauf

Abbildung 1.1 zeigt, wie der IslEnquirer im Laufe der Zeit und mit weiteren Sessions immer neue Personen kennenlernt, mit denen er dann in späteren Sessions teilweise erneut spricht. Die dabei gelernten Namen

bzw. Label sind jedoch nicht immer richtig, so dass mit der Zeit eine immer größere werdende Divergenz zwischen Anzahl gelernter Namen (in der Abbildung mit der Farbe Türkis eingezeichnet) und Anzahl getroffener Personen (in der Abbildung mit der Farbe Magenta eingezeichnet) entsteht. Mit der dritten Kurve (in der Abbildung grün als inverse Session Error Rate eingezeichnet) wird prozentual die Anzahl korrekt gelabelter Sessions der Gesamtzahl aufgezeichneter Sessions gegenübergestellt, deren Wert gegen Ende der Grafik auf knapp 70% fällt. Das bedeutet, dass gegen Ende des betrachteten Zeitraums 30% aller Sessions in der Wissensbasis mit einem falschen Label gespeichert vorliegen.

Möchte man der Tendenz eine sich immer weiter verunreinigenden Wissensbasis entgegentreten scheint eine Analyse der Daten verbunden mit einem zielgerichteten Korrigieren oder Vergessen wertloser bzw. schädlicher Information der einzige Ausweg. Insbesondere, wenn das System, wie es die Ziele des IslEnquirer vorsehen, über einen längeren Zeitraum autonom agieren soll.

1.2 Überblick

Die vorliegende Arbeit unterteilt sich in mehrere Abschnitte, die jeweils unterschiedliche Schwerpunkte setzen. Kapitel 2 bietet einen Überblick über grundlegende Verfahren, Algorithmen und Methoden, die in dieser Arbeit zum Einsatz kamen beziehungsweise ihre Grundlage bilden. Kapitel 3 stellt verwandte Arbeiten vor und grenzt sie gegenüber dem von uns vorgeschlagenen Ansatz ab. Kapitel 4 stellt den Kern der Arbeit dar und beschreibt zum einen die Fehlerdetektion und zum anderen die Fehlerbereinigung auf der Wissensbasis. Kapitel 5 befasst sich mit den Daten und beschreibt die durchgeführten Ergebnisse und Evaluationen. Kapitel 6 fasst die Ergebnisse noch einmal zusammen und zeigt Ansätze für zukünftige Arbeiten auf.

2 Terminologie und Grundlagen

Das folgende Kapitel schafft einen Überblick über die verschiedenen Komponenten, Methoden und Begriffe, die in der vorliegenden Arbeit zum Einsatz kamen bzw. ihre Grundlage bilden. Das Kapitel untergliedert sich dabei in mehrere Teile.

Im ersten Teil des Kapitels wird der IslEnquirer eingeführt, der mit seinen Aufgaben und Zielen das Szenario der hier vorgestellten Arbeit stellt. Die zwei darauf folgenden Abschnitte befassen sich mit zwei wichtigen Aspekten des IslEnquirers, die auch für die vorliegende Arbeit bedeutsam sind. Dabei wird zunächst der Aufbau eines natürlichsprachigen Dialogsystems beschrieben und anschließend, wie der IslEnquirer die Identität seines Gegenübers ermittelt. Im anschießenden vierten Teil wird die Wissensbasis des IslEnquirers erläutert und welche Typen von Fehlern sich auf ihr unterscheiden lassen. Dieses Unterkapitel ist für die vorliegende Arbeit insofern zentral, da es die zu lösenden Probleme beschreibt, die auf der Wissensbasis existieren.

Die abschießenden drei Teile beschreiben, von dem Szenario unabhängig, Methoden bzw. Vorgehensweisen, auf die wir in späteren Kapiteln zurückgreifen werden. Der fünfte Teil befasst sich mit grundlegenden Techniken der Bildverarbeitung, der sechste Teil mit der logistischen Regression und der siebte und abschießende Teil stellt verschiedene Distanzmaße vor.

2.1 IslEnquirer

Das dieser Arbeit zugrundeliegende System besteht aus einem Stereokamerasystem mit schwenkbarem Kopf, jeweils einem Nah- und Fernsprechmikrophon und Lautsprechern. Veranschaulicht wird der Aufbau in Abbildung 2.1. Diese Komponenten sind mit einem Computer verbunden, auf welchem sowohl die Verarbeitung der Audio- und Videodaten, als auch die Steuerung des Dialogs durchgeführt werden. Zur Steuerung des Dialogs wird der Dialogmanager Tapas [Hol08] verwendet, auf dessen Grund-

lage Felix Putze in seiner Diplomarbeit [Put08] einen modularen Ansatz einer Dialogstrategie, basierend auf dem Entwurf von Hartwig Holzapfel [HW07], implementiert hat. Der Namenlerndialog wird dabei als ein Modul des Systems aufgerufen. Die Spracherkennung wird durch „Janus“ und die Sprachausgabe mit der Software Swift realisiert. Zur Personenidentifikation werden sowohl FaceID als auch VoiceID herangezogen, die, wie in [GHW08] vorgestellt, zu einer gemeinsamen MultimodalenID vereinigt werden.

Abbildung 2.1: Der IslEnquirer im Gespräch. Entnommen: [Put08]

Ziel des IslEnquirers ist es mittels Dialoge soziale Informationen über Personen zu sammeln, die am ISL arbeiten [Put08]. Die gewonnenen Informationen werden genutzt, um soziale Netzwerke, wie beispielsweise Arbeitsgruppen oder Rollen, welche die Personen in den Netzwerken einnehmen, zu erkennen. Die resultierende interne Struktur des Instituts wird auf einer Internetseite abgebildet. Dabei werden Änderungen, die während der Laufzeit des Systems entstehen, dynamisch übernommen.

Die Interaktion mit verschiedenen Benutzern wird in Sessions aufgeteilt. Eine Session entspricht dabei der Interaktion mit genau einem Benutzer. Um dies modellieren zu können wird ein Session-Modell verwendet, das den Zustand der Sessions und der Transitionen zwischen diesen Zuständen behandelt. Dabei wird bei einer Benachrichtigung über den Beginn einer Session die Aufnahme gestartet und mit der Personenidentifikation begonnen, bis eine Nachricht eintrifft, dass die Session beendet werden

kann. Beispielsweise geschieht dies, wenn über einen bestimmten Zeitraum hinweg keine Eingabe vom Benutzer erkannt werden konnte und demnach angenommen werden muss, dass der Benutzer den Dialog abgebrochen hat. Eine neue Session wird initialisiert, wenn entweder der Personentracker eine Person erkannt hat oder eine Spracheingabe erkannt wurde. Um das Auslösen einer neuen Session direkt nach dem Beenden des Dialoges zu verhindern, wird dies für einige Sekunden unterdrückt. Dadurch ist es dem Benutzer möglich das System zu verlassen ohne erneut in einen Dialog verwickelt zu werden. Außerdem kann in dieser Zeit das System die neue Person für seine Face- und VoiceID eintrainieren und andere interne Operationen ausführen, die nicht online geschehen können.

2.2 Natürlichsprachige Dialogsysteme

Ein natürlichsprachiges Dialogsystem oder englisch „Spoken Dialog System" setzt sich aus mehreren Komponenten zusammen, die in einer vorgegebenen Reihenfolge die Eingabe verarbeiten. Diese Kette von Modulen wird in jedem Turn (einzelne Benutzereingabe) durchlaufen, um zu jeder Benutzereingabe die passende Ausgabe des Systems zu erzeugen. Die wichtigsten Komponenten sind dabei die Spracherkennung, die NLU-Einheit[1], der Dialogmanager und das Sprachsynthesemodul.

Die automatische Spracherkennung erzeugt aus der mittels Nahsprechmikrophon eingegebenen Sprache eine Textrepräsentation des Gesprochenen. In der NLU-Einheit wird der erzeugte Text in eine semantische Repräsentation des Gesprochenen überführt. Diese hat die Form einer typisierten Merkmalstruktur (Typed Feature Structure, TFS) [Car92]. Bei mehreren Eingabemodalitäten existiert für jede dieser Modalitäten eine eigene Einheit zur semantischen Interpretation der Eingabe. Diese TFS wird anschließend im Kontext des aktuellen Dialogzustandes interpretiert. Der Dialogzustand setzt sich hierzu aus der Gesamtheit an bisherigen Äußerungen, dem Diskurs und internen Variablen zusammen. Bei dieser Interpretation können auch Konvertierungsregeln angewendet werden, um die Eingabe TFS auf die Erwartung des Dialogsystems abzubilden. Auf Grundlage der daraus resultierenden semantischen Repräsentation wählt die Dialogstrategie die nächste Systemaktion aus (Move). Dabei ist ein Move eine abstrakte Repräsentation und beinhaltet mehrere Einzelaktionen, wie zum Beispiel Systemausgaben oder Datenbankzugriffe. Veranschaulicht wird die Modulkette durch eine schematische Darstellung in Abbildung 2.2.

[1] Natural Language Understanding: „Verstehen natürlicher Sprache"

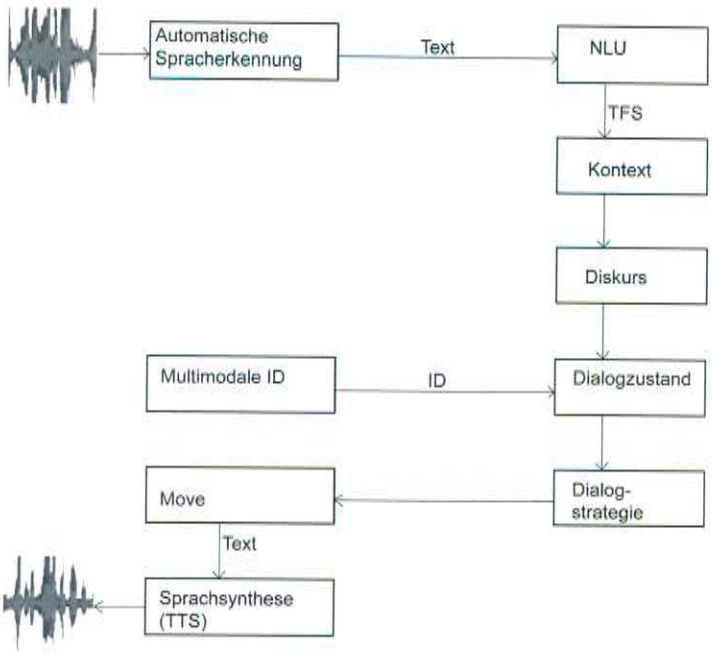

Abbildung 2.2: Aufbau eines Dialogsystems. Ähnlich in: [Put08]

Die Kommunikation zwischen den einzelnen Modulen wird über einen Kommunikationsagenten realisiert. Der IslEnquirer [Put08] und das Tapas Framework[Hol08] verwenden dazu das One4All-Protokoll[2].

2.2.1 Automatische Spracherkennung

Bei heutigen automatischen Spracherkennern funktioniert die Spracherkennung auf statistischer Basis. Dabei wird der wahrscheinlichste Kandidat $\hat{W}$ aller Wortsequenzen W geschätzt, wobei eine Sequenz von Merkmalsvektoren X gegeben ist.

$$\hat{W} = argmax_w P(W|X) = argmax_w P(X|W)P(W)$$

$P(X|W)$ repräsentiert das akustische Modell und ist die Wahrscheinlichkeit für das Auftreten eines bestimmten Merkmalsvektors X unter der Bedingung, dass ein Wort W aufgetreten ist. Dies wird meist mittels Hidden Markov Modellen (HMMs) für einzelne Phoneme modelliert. P(W) stellt das Sprachmodell dar. Es wird häufig mit Hilfe einer kontextfreien Grammatik modelliert, wobei alle Sätze, die nicht durch die Gramma-

[2]Ein multimodales System, entwickelt am ISL, Universität Karlsruhe

tik abgedeckt werden, eine Wahrscheinlichkeit von „Null" erhalten. Ein Vorteil der Modellierung mittels kontextfreier Grammatik ist, dass nur jene Hypothesen erkannt und an die NLU-Einheit weitergegeben werden, die für sie auch interpretierbar sind. Dies gilt allerdings nur, wenn beide Module die gleiche Grammatik verwenden. Eine weitere Möglichkeit das Sprachmodell zu modellieren, stellt das Konzept der n-Gramme dar, das in heutigen Spracherkennern häufig Anwendung findet. Dabei wird aufgrund eines Trainingskorpus für jede Wortfolge eine Wahrscheinlichkeit berechnet, mit der ein Wort auf eine Sequenz von n-1 vorangegangenen Wörtern folgt. Grundlegend für dieses Modell ist dabei die Markov-Annahme.

2.2.2 Verstehen natürlicher Sprache

Das Verstehen natürlicher Sprache bildet den Kern eines Dialogsystems. Es ist das Ziel dieses Moduls aus einer Benutzereingabe in Textform Semantik zu extrahieren. Dazu wird die Eingabe in ihre semantische Repräsentation transformiert, dargestellt durch eine typisierte Merkmalsstruktur (Typed Featured Structure, TFS) [Car92]. Eine solche TFS beinhaltet Merkmal-Wert-Paare und beschreibt mit ihnen die Eigenschaften eines semantischen Konzeptes. Dabei kann ein solches Attribut ein atomarer Wert oder erneut eine TFS sein. Jede TFS hat einen bestimmten Typ, der in einer Ontologie beschrieben ist. Diese Ontologie ist hierarchisch aufgebaut und unterstützt das Konzept der Vererbung. Durch die Darstellung der Semantik als TFS lässt sich die neue Eingabe auch in den Diskurs einbetten. Zum Verstehen der Sprache wird eine semantische kontextfreie Grammatik verwendet. In der Grammatik wird unterschieden zwischen Terminalen und Nichtterminalen. Die Nichtterminale teilen sich wiederum in syntaktische und semantische Einträge. Syntaktische Einträge dienen lediglich der Grammatikstruktur, wohingegen semantische Einträge ein Konzept der Ontologie widerspiegeln. Zusätzlich dazu enthält ein semantischer Eintrag eine grammatikalische Klasse und einen zusätzlichen, frei wählbaren Tag. Das Janus Framework und das Tapas Dialog Toolkit [Hol08], dessen NLU-Einheit in dem zugrundeliegenden System zum Einsatz kommt, nutzen dieselbe Grammatik.

2.2.3 Sprachsynthese

Da ein Dialog bekanntlich aus mindestens zwei Sprechern besteht, muss das Dialogsystem auch in der Lage sein mit dem Benutzer zu sprechen. Dafür wird ein Sprachsynthesewerkzeug verwendet, welches Text in Sprache

umwandelt (Text-to-speech system, TTS). Dies kann auf zwei verschiedene Arten geschehen. Bei einem konkatenativen Ansatz werden kurze Audiosegmente natürlicher Sprache aneinander gehängt. Werden die Übergänge zwischen den einzelnen Audiosegmenten noch geglättet und der Sprache Prosodie hinzugefügt, erhält man eine menschenähnliche Stimme. Nachteil dieses Verfahrens ist, dass hierbei eine große Menge Audiodaten benötigt wird. Der zweite Ansatz ist modellbasiert. Durch Training von Modellparametern kann eine künstliche Stimme erzeugt und angepasst werden, was den Vorteil birgt, dass bei diesem Vorgehen deutlich weniger Audiodaten benötigt werden. Jedoch wirkt die erzeugte Stimme maschinell und unnatürlich. In dieser Arbeit wird das Sprachsynthesewerkzeug Cepstral2 verwendet. Es unterstützt eine automatische Abbildung von Graphemen zu Phonemen, die mittels Entscheidungsbäumen trainiert wird. Zusätzlich dazu können Graphem-Phonem-Zuordnungen von Hand in ein Wörterbuch eingetragen werden. Dies ist besonders bei Namen hilfreich, da diese häufig anderen Regeln gehorchen als „normale“ Sprache. Zur Modifizierung der Sprachausgabe unterstützt Cepstral die Speech Synthesis Markup Language (SSML). Besondere Bedeutung hat dabei die Möglichkeit der Betonung einzelner Wörter oder Wortgruppen. Nebenprodukt der Betonung ist ein langsameres und deutlicheres Aussprechen, was insbesondere bei der Aussprache von Namen sehr hilfreich ist.

2.3 Die Identität des Gesprächspartners

Mit dem Ziel des IslEnquirers Informationen über Gesprächspartner zu sammeln, kommt der Erkennung bekannter Personen eine besondere Rolle zu. Denn wird eine bereits bekannte Person nicht als solche erkannt, können die durch den neuen Dialog akquirierten Informationen nicht richtig zugeordnet werden und die Wissensbasis wird „verunreinigt“. Genau diese Art der „Verunreinigung“ der Wissensbasis bildet die Grundlage der hier vorliegenden Arbeit. Bevor wir uns jedoch in Kapitel 4 mit der Lösung dieses Problems befassen, sollen im Folgenden die zwei Komponenten vorgestellt werden, die bei der Personenerkennung maßgeblich beteiligt sind.

Die erste hier zu nennende Komponente ist die MultimodaleID [GHW08], deren Aufgabe es ist, auf Grundlage der im Dialog entstehenden Bild- und Audioseuquenzen möglichst schnell und zuverlässig eine Hypothese mit zugehöriger Konfidenz bereitzustellen, ob das Gegenüber des Robotors eine bekannte Person ist, und wenn, um wen es sich handelt. Die zweite maßgebliche Komponente ist der in Kapitel 2.1 erwähnte Namen-

lerndialog, dessen Aufgabe es ist, in einem möglichst natürlich wirkenden Dialog den Namen des Gesprächspartners in Erfahrung zu bringen. Um dies zu gewährleisten und nicht unnötig bekannte Personen immer wieder erneut nach ihren Namen zu fragen, wird zu Beginn des Dialogs auf die Ergebnisse der MultimodalenID zugegriffen und geprüft, ob diese eine Hypothese mit hoher Konfidenz bereitstellt. Ist dies der Fall, kann der Namenlerndialog dahingehend vereinfacht werden, dass lediglich das von der Multimodalen-ID zur Verfügung gestellte Label bzw. der Name abgefragt wird, so dass in diesem Fall der normale Namenlerndialog nur dann durchlaufen werden muss, wenn die MultimodaleID sich in ihrer Hypothese geirrt haben sollte und der Gesprächspartner den vorgeschlagen Namen zurückweist. Dieses Vorgehen hat neben dem Ziel eines natürlicher wirkenden Dialogs auch den Vorteil, dass die Erkennung einer einfachen Bestätigung eines bereits bekannten Namens deutlich robuster gelingt, als die Erkennung des freigesprochenen Namens. Insbesondere bei Nachnamen, die in der Regel nicht im Vokabular des Spracherkenners vorliegen, ist man beim Lernen von neuen Namen häufig gezwungen auf das Mittel der Buchstabierung zurückzugreifen, was einem flüssigen und damit natürlich wirkenden Dialog entgegensteht. Wurde die Person nicht bereits durch die MultimodaleID erkannt und im Dialog bestätigt, muss auch nach dem Erfragen des Namens durch den Namenlerndialog der Name anschließend bestätigt werden. Dieser Bestätigungsschritt soll verhindern, dass missverstandene Namen unkontrolliert in die Wissensbasis gelangen. Wenngleich es auch hier zu so genannten „Fehlbestätigungen" kommen kann, so dass am Ende des Dialogs ein teilweise oder aber vollkommen falscher Name in die Wissensbasis gelangt. Ein Beispiel für einen Namenlerndialog bei einer unbekannten Person findet sich in Tabelle 2.1.

System:	I am Robbi Robinson, what is your name?
User:	My name is Philipp Große
System:	Did you say, your first name is Philipp?
User:	Yes
System:	However, I didn't get your last name, can you please repeat it?
User:	My last name is Große
System:	Could you please spell that?
User:	G R O S S E
System:	Did you say, your last name is Grosse?
User:	Yes
System:	Nice to meet you, Philipp

Tabelle 2.1: Beispieldialog bei unbekanntem Benutzer. Ähnlich in [Hol09]

2.4 Wissensbasis

Als Wissensbasis oder Knowledgebase wird die Gesamtheit der gesammelten Daten verstanden, wobei hier insbesondere die - hoffentlich richtige - Zuordnung der gesammelten Informationen aus den Dialogen zu den jeweiligen Personen ausschlaggebend dafür ist, ob die Wissensbasis als „sauber", also richtig, oder aber „verunreinigt", also fehlerhaft, bezeichnet werden kann. Die ideale Wissensbasis würde für jede (dem System) bekannte Person alle (für das System) relevanten Informationen beinhalten, ohne dabei gleichzeitig so genannte Falscheinträge zu enthalten, die beispielsweise nicht existierende Personen beschreiben.

Wie in Kapitel 2.1 bereits angesprochen, werden die Daten des IslEnquirers für einen durchgeführten Lerndialog jeweils sessionweise gebündelt aufgezeichnet und abgespeichert. Diese Daten umfassen Dialog-Logdateien, die unter anderem den Namen bzw. das Label des jeweiligen Gesprächspartners beinhalten, und kurze Audiosegmente in Form von WAV Dateien, aber auch die für diese Arbeit wichtigen Einzelbilder (8-15 Frames pro Sekunde), die als JPEGs der Auflösung 640×480 Pixel mit 24-bit Farbtiefe gespeichert werden.

Die sessionweise Aufteilung und die Systemannahme, während eines Dialogs stets nur mit genau einer Person zu sprechen, gewährleisten, dass die Zuordnung des Namens des Gesprächspartners über sämtliche in einer Session aufgezeichneten Daten generalisiert werden kann. Umgekehrt bedeutet es aber auch, dass Analysen über die Beschaffenheit der, in einer Session gebündelten, Daten Rückschlüsse auf die Person ermöglichen. Genau diesen Umstand nutzen wir, wenn wir im Folgenden einen Cluster-Algorithmus verwenden, um Unstimmigkeiten in der Wissensbasis ausfindig zu machen. Unstimmigkeiten heißt hier vor allem, dass die aufgezeichneten Videobilder nicht mit dem Label für die Session übereinstimmen, wobei sich das Label direkt aus dem Namenlerndialog und dem darin bestätigten Namen ableitet. Verunreinigungen der Wissensbasis entstehen also vor allem durch die in Kapitel 2.3 erwähnten „Fehlbestätigungen" von Namen.

2.4.1 Fehlertypen

Es lassen sich, wie in Grafik 2.3 dargestellt, im Wesentlichen vier Arten von Fehlern in der Wissensbasis unterscheiden.

Bei den ersten beiden Typen von Fehlern handelt es sich um so genannte Doppel-Einträge (*Duplikate*) in der Wissensbasis. Hier wird eine Person auf zwei Identitäten in der Wissensbasis abgebildet mit dem Effekt, dass

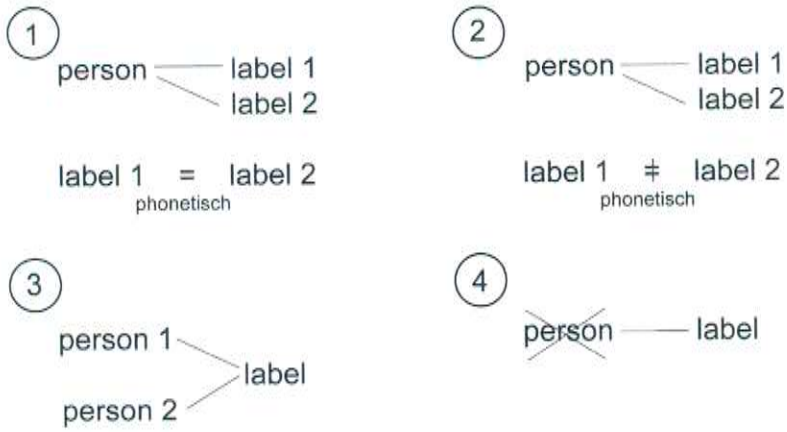

Abbildung 2.3: Vier unterscheidbare Fehlertypen auf der Wissensbasis

die im Dialog gesammelten Informationen nicht sinnvoll verknüpft werden können und im schlimmsten Fall sogar einen negativen Einfluss auf spätere Interaktionen mit dem System haben können.

Der erste Typ unterscheidet sich vom zweiten Typ vor allem darin, dass die beiden vom System verwendeten Labels phonetisch für den Benutzer nicht unterscheidbar sind. Zum Beispiel: Ein und dieselbe Person liegt einmal mit dem Namen „Stefan" und ein weiteres mal mit dem Namen „Stephan" in der Wissensbasis vor. Für den Benutzer ist diese Art des Fehlers nicht zu erkennen und eine Fehlbestätigung in einem natürlichen Dialog, in dem man auf eine explizite Buchstabierung für die Bestätigung verzichtet, unvermeidbar. Klassischerweise entsteht diese Art von Fehler, wenn der Name des Benutzers einmal im konventionellen Dialog gelernt wurde und ein weiteres Mal durch explizite Buchstabierung, wobei im ersten Fall vom System eine andere Schreibweise angenommen wurde. Folge dieses Typus von Fehlern in der Wissensbasis ist, dass das im Dialog gesammelte Wissen auf zwei separate Identitäten abgebildet wird. Da es sich bei den Video- und Audiodaten jedoch um dieselbe Person handelt ist es im Zweifelsfall reiner Zufall, für welches der Identitäten sich die MultimodaleID entscheidet, und je nach dem, wird mal das eine, mal das andere Modell um Informationen aus einem neuen Dialog vervollständigt, mit dem Effekt, dass die Person mehrfach die gleichen Fragen beantworten muss. Streng genommen handelt es sich bei mindestens 50% der Fälle auch um eine Fehlklassifikation der MultimodalenID, da schließlich bestenfalls eines der Labels der korrekten Schreibweise entsprechen kann. Da

diese Fehlererkennungen vom Benutzer jedoch unentdeckt bleiben, kann dies beim ersten Fehlertyp vernachlässigt werden.

Anders ist es hingegen bei dem zweiten Typus von Fehlern, bei dem die verwendeten Label phonetisch eindeutig unterschiedlich sind. Beispielsweise hätte „Stephan“ in diesem Fall versehentlich „Stebha“ als seinen Name bestätigt. In diesem Fall erkennt die MultimodaleID zwar möglicherweise die richtige Person, wenn sie „Stebha“ vorschlägt, doch da das Label vom Benutzer nicht als sein Name wiedererkannt und somit auch nicht erneut bestätigt wird, können die zuvor gesammelten Informationen nicht genutzt werden. Mit diesem zweiten Fehlertyp wird vor allem die Brauchbarkeit der MultimodalenID untergraben, denn selbst wenn die MultimodaleID eine 100%ige Erkennungsrate hat, so ist sie doch auch auf die Zuverlässigkeit der verwendeten Label angewiesen. Eine Person, die eigentlich zu 100% richtig erkannt wird, jedoch mit drei verschiedenen Labeln in der Wissensbasis vertreten ist, hat so, vorausgesetzt eines der drei Label entspricht dem richtigen Namen, plötzlich eine a-priori-Wahrscheinlichkeit von 33,3% tatsächlich richtig erkannt zu werden. Somit entstehen zwei separate Probleme durch den zweiten Fehlertyp: Zum einen wird dem Benutzer zu Beginn einer neuen Session immer wieder ein im Zweifelsfall vollkommen unsinniger Name vorgeschlagen, was auf Dauer ausgesprochen lästig sein kann und keinen guten Eindruck des Systems hinterlässt. Zum anderen sind die unter dem fehlbestätigten Label gesammelten Informationen nicht mehr weiter brauchbar, denn eine Person mit diesem Namen wird kein zweites Mal mit dem System reden.

Der dritte Fehlertyp hat die destruktivste Auswirkung auf die Wissensbasis. Hierbei werden zwei verschiedene Personen auf das gleiche Label gemappt, mit der Konsequenz, dass die Daten untrennbar miteinander vermengt werden. Dies betrifft nicht nur die in den Dialogen gesammelten Informationen, bei denen ursprünglich richtige Einträge möglicherweise überschrieben oder verfälscht werden, sondern auch Audio- und Videodaten, auf deren Basis die MultimodaleID klassifiziert. Ähnlich wie schon bei dem zweiten Fehlertyp erkennt die MultimodaleID zwar möglicherweise die richtige Person, bezieht sich dabei jedoch auf das fehlerhafte Label und kann von dem Benutzer nur zurückgewiesen werden. Theoretisch kann dieser Fehler auftreten, wenn zwei Personen mit dem gleichen Namen mit dem System reden und so auf dasselbe Label gemappt werden. Praktisch entsteht dieser Fehler jedoch, wie schon bei den vorangegangenen Fehlertypen, durch Fehlbestätigung; wenn das System mit einer neuen Person spricht, die MultimodaleID jedoch statt „Unknown“ eine bekannte Person meint erkannt zu haben, der Namenlerndialog daraufhin das von

der MutlimodalenID vorgeschlagene Label als Namen anbietet und es bei der darauf folgenden Bestätigung zu Missverständnissen kommt. Entweder weil ein „Nein" vom Spracherkenner als „Ja" gedeutet wird oder aber weil der Benutzer die Frage missverstanden hat.

Der vierte Fehlertyp in der Grafik 2.3 entsteht, anders als die ersten drei, nicht durch eine Fehlbestätigung, sondern durch die Veränderung der so genannten *Groundtruth*. Das heißt, eine Person die zuvor mit dem System geredet hat und in der Wissensbasis korrekt abgebildet wurde, gehört nicht mehr in die Wissensbasis. Beispielsweise soll das System alle Mitarbeiter eines Instituts kennen, aber einer der kennen gelernten Mitarbeiter hat das Institut inzwischen verlassen und arbeitet mittlerweile andernorts. In diesem Fall entspricht die Wissensbasis nicht (mehr) der Groundtruth.

Über die vier hier bereits vorgestellten Fehlertypen hinaus ließe sich noch ein fünfter Fehlertyp ausmachen, nämlich die Unvollständigkeit der Wissensbasis. Wenn also Personen der Groundtruth nicht in der Wissensbasis abgebildet werden, da das System mit ihnen noch nicht gesprochen hat. Dieser Fehlertyp wird in der vorliegenden Arbeit jedoch vernachlässigt, da es sich hierbei um keinen Fehler auf der Wissensbasis im eigentlichen Sinne handelt, sondern vielmehr um ein Problem der Wissensakquirierung.

2.5 Vektordarstellung von Bilddaten

Wie bereits in Kapitel 2.4 angedeutet, spielt das Clustern auf den sessionweise gebündelten Bilddaten für die hier vorgestellte Arbeit eine zentrale Rolle. Da Clustering-Algorithmen (siehe 3.1) jedoch in der Regel von einer Vektordarstellung der Daten ausgehen, müssen die als JPEGs vorliegenden Bilddaten zunächst in eine solche überführt werden. Dazu wird für jedes Bild mit Hilfe des, von Viola und Jones entwickelten, Haar Kaskaden Klassifikators [VJ03] entschieden, ob auf dem Bild ein Gesicht zu finden ist und, wenn ja ein möglichst geschickter Bildausschnitt gewählt, so dass das Gesicht zentriert erfasst wird. Dieser 64x64 Pixel umfassende Gesichtsausschnitt des Bildes wird anschließend in 8x8 Blöcke unterteilt. Für jeden dieser 64 8x8 Pixel Blöcke wird eine DCT berechnet und mit Hilfe des Zig-Zag-Scans in eine Vektordarstellung überführt, wobei letztlich nur die ersten fünf, genauer der zweite bis sechste, Koeffizient für den finalen Gesichtsausschnittsvektor verwendet wird. Der erste DCT-Koeffizient ist in sofern uninteressant, als er nur den durchschnittlichen Grauwert

des Bildes wiedergibt. Der finale Bildvektor umfasst damit $64 \times 5 = 320$ Koeffizienten, die das beinhaltete Gesicht repräsentieren.

2.5.1 Diskrete Kosinustransformation

Die diskrete Kosinustransformation (DCT) ist eine lineare, orthogonale Transformation, die ein zeitdiskretes Signal in den Frequenzbereich transformiert. Seit 1974 ist sie die am weitesten verbreitete Transformation zur Redundanzreduktion von Bildsignalen [Wik09]. Die Gründe dafür sind vielfältig. Einerseits lässt sich die DTC sowohl in Software wie auch in Hardware implementieren. Andererseits transformiert die DCT Bilddaten effektiv in eine Form, die gut komprimiert werden kann. Im Gegensatz zur diskreten Fouriertranformation rechnet man bei der Kosinustransformation nicht mit komplexen, sondern lediglich mit reellen Koeffizienten, was einen weiteren Vorteil darstellt.

Bei der Gesichtsidentifikation interessiert man sich ausschließlich für die Transformation vom Bildbereich in den Frequenzbereich, die im Folgenden mit FDCT[3] abgekürzt wird. Um Korrelationen in horizontaler und vertikaler Bildrichtung zu erfassen, wird die zweidimensionale Variante der FDCT benutzt. Zu diesem Zweck wird das Bild in Blöcke von 8x8 Pixel zerlegt. Die folgende Gleichung beschreibt die zweidimensionale FDCT für einen 8x8-Block eines Bildes:

$$F_{x,y} = \frac{2C(x)C(y)}{8} \sum_{i=0}^{7} \sum_{j=0}^{7} f_{i,j} \cdot cos\left(\frac{(2i+1)x \cdot \pi}{16}\right) \cdot cos\left(\frac{(2j+1)y \cdot \pi}{16}\right)$$

wobei $f_{i,j}$ den Wert des Punktes (i, j) im Eingabebild darstellt, $F_{x,y}$ die DCT-Koeffizieten an der Stelle (x, y) und $C(x)$ und $C(y)$ die Konstanten.

$$C(z) = \begin{cases} \frac{1}{\sqrt{2}} & , z{=}0 \\ 1 & , z{\neq}0 \end{cases}$$

Die FDCT repräsentiert jeden Block eines Bildausschnittes durch gewichtete Summen von 2D-Kosinusfunktionen, auch Basisfunktionen genannt. Das Muster oben links hat die niedrigste Frequenz und ist nur ein Einheitsblock. Von links nach rechts nimmt die "Frequenz" zwischen hell und dunkel in horizontaler Richtung zu. Von oben nach unten nimmt hingegen die "Frequenz" zwischen hell und dunkel in vertikaler Richtung zu.

[3] forward discrete cosinus transformation

Folglich nehmen sowohl die horizontalen als auch die vertikalen "Frequenzen" in diagonaler Richtung gleichzeitig zu. Abbildung 2.4 verdeutlicht diesen Zusammenhang:

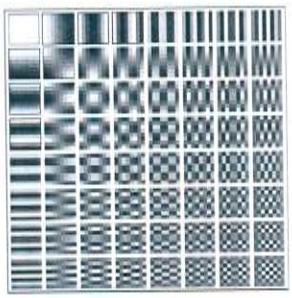

Abbildung 2.4: Basis der Diskreten Kosinustransformation für 8x8 Pixel. Entnommen: [Wik09]

2.5.2 Zig-Zag-Scan

Durch die DCT erhält man für jeden 8x8 Pixelblock 64 Koeffizienten. Der sogenannte Zig-Zag-Scan sortiert diese nun dahingehend um, dass sie in einem eindimensionalen Vektor dargestellt werden können. Er beginnt dabei in der linken oberen Ecke und bewegt sich im zickzack-Kurs zur rechten unteren Ecke. Folglich stehen die niedrigfrequenten Koeffizienten anschließend am Anfang des Vektors. Veranschaulicht wird dieses Vorgehen durch Abbildung 2.5.

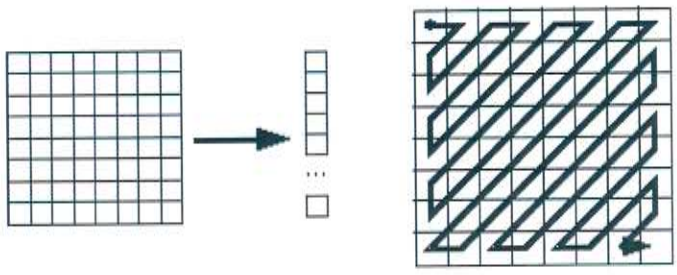

Abbildung 2.5: Der Zig-Zag-Scan ordnet die Koeffizienten in einen eindimensionalen Vektor an. Entnommen: [Zig09]

2.6 Logistische Regression

Unter logistischer Regression versteht man ein Verfahren zur multivarianten Analyse des Zusammenhangs zwischen binär abhängigen Variablen und mindestens einer unabhängigen Variablen [LRZ09]. Typische Beispiele für binäre abhängige Variablen sind Variablen, die das Eintreten eines Ereignisses erfassen. Diese Variablen haben nur zwei mögliche, sich gegenseitig ausschließende Ausprägungen, wie z.B. „Ereignis findet statt" ($Y = 1$) und „Ereignis findet nicht statt" ($Y = 0$). Nun interessiert uns der Einfluss der jeweiligen unabhängigen Variablen auf die Eintrittswahrscheinlichkeit. Die logistische Regression löst diese Aufgabe durch eine geeignete Transformation der abhängigen Variablen. Sie geht von der Idee der Odds aus, d.h. dem Verhältnis von $P(Y = 1)$ zur Gegenwahrscheinlichkeit $1 - P(Y = 1)$ bzw. $P(Y = 0)$.

$$Odds(Y = 1) = \frac{P(Y = 1)}{1 - P(Y = 1)} = \frac{P(Y = 1)}{P(Y = 0)}$$

Die Odds können zwar Werte >1 annehmen, doch ist ihr Wertebereich nach unten beschränkt, da sie sich asymptotisch Null annähern. Ein unbeschränkter Wertebereich wird durch die Transformation der Odds in die sog. Logits

$$Logit(Y_{1/0}) = ln\frac{P(Y = 1)}{P(Y = 0)}$$

erreicht, die Werte zwischen minus und plus unendlich annehmen können. In der logistischen Regression wird dann die Regressionsgleichung

$$Logit(Y_{1/0}|X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n$$

geschätzt. Es werden also die Regressionkoeffizienten β_i (auch Logit-Koeffizienten) für jede unabhängige Variable X_i bestimmt, nach denen die Logits für ein gegebenes X berechnet werden können.

Durch zwei Transformationsschritte lassen sich die Einflüsse der logistischen Regression auch als Einflüsse auf die Eintrittswahrscheinlichkeit P(Y=1) ausdrücken:

$$P(Y = 1|X) = \frac{e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\cdots+\beta_n X_n}}{1 - e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\cdots+\beta_n X_n}}$$

Insbesondere aufgrund dieser Interpretation haben wir uns für den Einsatz der logistischen Regression entschieden, denn sie ermöglicht uns mit

Hilfe verschiedener Konfidenzmerkmale als X_i die Eintrittswahrscheinlichkeit $P(Y = 1)$ zu schätzen, welche wiederum ein ideales Konfidenzmaß darstellt, das wir, wie wir noch zeigen werden, als adaptives Distanzmaß für das Clustering nutzen können.

2.7 Distanzmaße

Die meisten Clustering-Verfahren beruhen darauf Distanzen zwischen den zu gruppierenden Objekten zu nutzen, um eben jene Gruppen oder Cluster zu erkennen bzw. die Ähnlichkeit zwischen den Gruppen auszumachen. Dazu steht eine Vielzahl an möglichen Distanzmaßen zur Verfügung, die je nach Art und Verteilung der zu gruppierenden Objekte, Vorteile und Nachteile bieten. Im Folgenden sollen einige der gängigsten Distanzmaße vorgestellt werden.

Klassischerweise werden vier verschiedene Vorgehensweisen für die Ähnlichkeits- bzw. Distanzbestimmung zweier Cluster unterschieden.

1. **Single Link**: Distanz zwischen den beiden nächsten Punkten der Cluster.
2. **Complete Link**: Distanz zwischen den beiden entferntesten Punkten der Cluster.
3. **Group Average Link**: Durchschnittliche Distanz zwischen den Clustern.
4. **Centroid**: Distanz zwischen mittleren Repräsentanten der Cluster.
   a) Verwendung des Punktes am nächsten zum Zentrum des Clusters
   b) Verwendung des Durchschnittsvektoren des Clusters

Der Aufwand für die ersten drei Verfahren is vor allem dadurch bestimmt, dass hierzu jeder Punkt des einen Clusters mit jedem des anderen Clusters verglichen werden muss, was folglich auf einen Rechenaufwand von $O(n_1 \cdot n_2)$ hinausläuft, während ein centroidbasierter Ansatz nur einen Aufwand von $O(n_1 + n_2)$ für die Bestimmung der Durchschnittsvektoren benötigt. Die beiden Variablen n_1 und n_2 bezeichnen dabei die Anzahl von Punkten bzw. Vektoren pro Cluster.

Unabhängig davon für welches der vier Vorgehensweise man sich entscheidet, ist die Distanz zwischen den Clustern vor allem durch die Distanz ihrer Punkte bzw. Vektoren bestimmt. Hierzu wird in der Regel die Euklid-Distanz verwendet, wahlweise kommt jedoch auch die Manhattan-Distanz zum Einsatz. Diese sind wie folgt definiert:

- **Manhattan-Distanz**

$$D_1 = \left|\vec{X_1} - \vec{X_2}\right| = \sum_{i=1}^{d} \left|\vec{X_1}^{(i)} - \vec{X_2}^{(i)}\right| \tag{2.1}$$

- **Euklid-Distanz**

$$D_2 = \sqrt{(\vec{X_1} - \vec{X_2})^2} \tag{2.2}$$

Das in dieser Arbeit als *durchschnittliche inter-Cluster-Distanz* bezeichnete Distanzmaß entspricht der Idee beim *Group Average Links*. Sie ist definiert durch:

- **Durchschnittliche inter-Cluster-Distanz**

$$D_3 = \sqrt{\frac{\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} (\vec{X_i} - \vec{X_j})^2}{n_1 \cdot n_2}} \tag{2.3}$$

Neben der Distanz zwischen den Clustern gibt es beim Clustering noch eine weitere wichtige Distanz. Die Distanz oder Varianz innerhalb eines Clusters, häufig als *intra-Cluster-Distanz* bezeichnet. Doch auch auf Basis dieser lässt sich eine Distanz zwischen verschiedener Cluster bestimmen.

- **Durchschnittliche intra-Cluster-Distanz**

$$D_4 = \sqrt{\frac{\sum_{i=1}^{n_1+n_2} \sum_{j=1}^{n_1+n_2} (\vec{X_i} - \vec{X_j})^2}{(n_1 + n_2)(n_1 + n_2 - 1)}} \tag{2.4}$$

3 Verwandte Arbeiten

Dieses Kapitel gibt einen Überblick über Arbeiten, die sich mit ähnlichen Fragestellungen und Problemen befassen, wie die hier vorgestellte Arbeit. Hierzu unterteilt sich das Kapitel in drei thematischen Blöcke. Der erste Teil stellt verschiedene Clustering Methoden vor und grenzt sie gegen das in dieser Arbeit verwendete Vorgehen ab. Der zweite Teil befasst sich mit Active Learning und den Parallelen zu der vorliegenden Arbeit und der dritte Teil stellt klassische Ansätze zu Duplikateliminierung aus dem Bereich des Datenbank Cleansings vor.

3.1 Clustering

Unter Clusteranalyse oder Clustering versteht man strukturentdeckende, multivariate Analyseverfahren zur Ermittlung von Gruppen (Cluster) von Objekten, deren Eigenschaften bzw. Eigenschaftsausprägungen bestimmte Ähnlichkeiten aufweisen [HK06]. Die an verschiedene Anforderungen angepassten Verfahren der Clusteranalyse werden im Bereich der automatischen Klassifizierung, zur Erkennung von Mustern, im Bereich des Data-Mining oder im Information Retrieval eingesetzt. In Abgrenzung zu überwachten Lernverfahren sind die Klassenzugehörigkeiten der zu untersuchenden Objekte beim Clustering im Vorfeld nicht bekannt. Es handelt sich demnach bei Cluster-Algorithmen um eine Gruppe so genannter unüberwachter Lernverfahren.

Im Allgemeinen werden die zu untersuchenden Objekte beim Clustering als Punkte in einem Vektorraum dargestellt, deren Dimensionen die Merkmalsausprägungen bilden. Ein Cluster ist eine Anhäufung von Punkten (Punktwolke), wobei die Abstände zwischen den Punkten zueinander oder die Varianz innerhalb eines Clusters als Proximitätsmaße dienen. Alternativ können Cluster auch als Gruppe von Objekten definiert werden, die in Bezug auf einen berechneten Schwerpunkt eine minimale Abstandssumme haben. In diesem Fall ist die Wahl eines geeigneten Distanzmaßes erforderlich.

Im Folgenden werden einige der klassischen Cluster-Methoden sowie dazugehörige Algorithmen vorgestellt. Dazu gehören unter anderem hierarchisches, partitionierendes, dichtebasiertes und modellbasiertes Clustering.

3.1.1 Partitionierendes Clustering

Beim partitionierenden Clustering wird die Datenbasis, bestehend aus n Objekten, in eine zuvor festgelegte Anzahl von k Partitionen ($k \leq n$) unterteilt, wobei jede Partition einem Cluster entspricht. Anschließend an diese initiale Partitionierung werden die Objekte mit einem iterativen Umsortierungsmechanismus zwischen den Clustern umverteilt, mit dem Ziel, dass die Objekte in einem gemeinsamen Cluster möglichst ähnlich und Objekte in verschiedenen Clustern möglichst verschieden zueinander sind. Um bei einem partitionierenden Clustering die global optimale Lösung zu finden, müssten theoretisch alle möglichen Datenpartitionierungen durchlaufen werden. Stattdessen arbeiten die meisten dieser Verfahren mit heuristischen Methoden, wie etwa mit Hilfe von Durchschnittsvektoren, im Fall des k-Means-Algorithmus. Bei komplexeren Clusterformen wird ein solches heuristisches Vorgehen bei partitionierenden Cluster-Methoden zum Nachteil.

3.1.1.1 k-Means

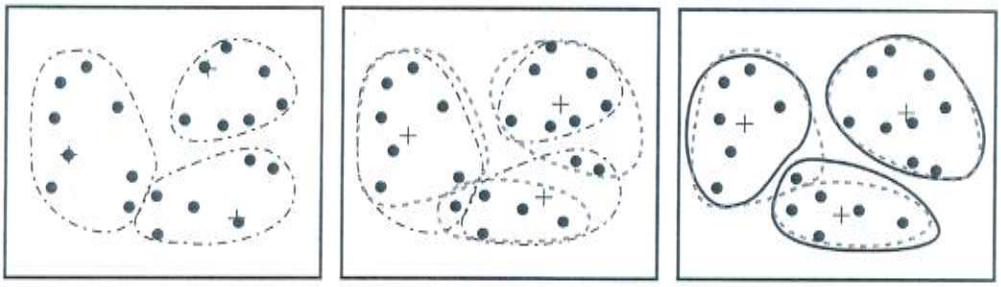

Abbildung 3.1: Clustering mit k-Means. (Der Durchschnittsvektor wird durch „+“ dargestellt.) Entnommen: [HK06]

Der wohl bekannteste und am häufigsten verwendete Vertreter des partitionierenden Clusterings ist der k-Means-Algorithmus. Hierbei werden die zu gruppierenden Objekte derart in k Partitionen unterteilt, dass die

resultierende intra-Cluster Ähnlichkeit möglichst hoch und die inter-Cluster-Ähnlichkeit möglichst gering ist, wobei die Ähnlichkeit eines Objekts anhand der Distanz zwischen dem Objekt-Vektor und dem Durchschnittswert (oder *Centroiden*) der Vektoren des jeweiligen Clusters bemessen wird. Der k-Means-Algorithmus geht dabei wie folgt vor: Zunächst werden zufällige k Objekte als initiale Cluster-Centroiden gewählt. Alle restlichen Objekte werden dem jeweils ähnlichsten Cluster zugeordnet, wobei hier die Distanz zwischen dem jeweiligen Cluster-Centroiden und dem Objektvektor zugrunde gelegt wird. Nachdem alle Objekte einem Cluster zugeordnet wurden, werden für alle Cluster neue Durchschnittsvektoren berechnet. Dieses Vorgehen wird iterativ wiederholt bis ein Abbruchkriterium erfüllt ist. Typischerweise verwendet man hierzu den quadratischen Fehler als Abbruchkriterium

$$E = \sum_{i=0}^{k} \sum_{p \in C_i} |p - m_i|^2, \tag{3.1}$$

wobei p der Objekt-Vektor und m_i der Durchschnittsvektor für das Cluster C_i ist.

Der k-Means Algorithmus weist in seinen Grundzügen und der Idee mit Centroiden zu arbeiten einige Parallelen zu dem in dieser Arbeit verwendeten Ansatz auf. Die alternative Verwendung des k-Means auf DCT-Vektor-Ebene scheint in unserem Szenario jedoch wenig sinnvoll, da in diesem Fall die sessionweise Bündelung der Daten aufgehoben würde. Denkbar wäre die Anwendung des k-Means, ähnlich wie im hier vorgeschlagenen Ansatz, auf Session-Cluster-Ebene. Dann müsste jedoch die Anzahl an verschiedenen Personen in der Wissensbasis (sprich Information über die Groundtruth) bereits im Vorfeld des Clusterings bekannt sein, damit k als Parameter sinnvoll gewählt werden kann. Genau dieses Wissen fehlt uns in dem gegebenen Szenario eines autonom lernenden Dialog-Robotors.

3.1.2 Hierarchisches Clustering

Hierarchische Clustering-Methoden zeichnen sich insbesondere dadurch aus, dass sie, wie der Name schon andeutet, eine hierachische Struktur der Daten bzw. Cluster erzeugen. Dabei gibt es zwei entgegengesetzte Ansätze. Entweder man beginnt mit Clustern minimaler Größe und fusioniert diese schrittweise bis schließlich nur noch ein großes, alle Daten umfassendes, Cluster verbleibt. Diese Ansätze nennt man *Agglomerativ* oder

Bottom-up. Oder aber man beginnt bei einem alle Daten umfassenden Cluster, das man schrittweise in kleinere Cluster zerlegt. Diese Ansätze nennt man *Divisiv* oder *Top-down*. Das Vorgehen dieser beiden Ansätze wird anhand von fünf Objekten (a,b,c,d,e) in Abbildung 3.2 verdeutlicht. Die Schwächen hierarchischer Clustering-Methoden liegen zum einen in ihrer Skalierbarkeit und zum anderen in der sicheren Auswahl der richtigen Fusions- bzw. Zerlegungspunkte. Letzteres ist für solche Verfahren insofern besonders kritisch, da einmal durchgeführte Schritte (Fusionieren oder Zerteilen) nicht wieder rückgängig gemacht werden können und auch ein Verschieben innerhalb der Cluster nicht möglich ist. Um Fehlentscheidungen bei einzelnen Schritten zu vermeiden, und damit die Qualität des hierarchischen Clusterings zu sichern, gibt es zwei Ansätze: (1) Sorgfältige Analyse von Cluster „linkage" bei jedem hierarchischen Partitionieren, wie etwa bei Chameleon oder (2) agglomeratives Clustern von Objektgruppen in *Mikrocluster*, um diese dann mit einem zweiten Clustering-Verfahren bei einem *makroclustering* Schritt zusammenzufügen, wie bei BIRCH.

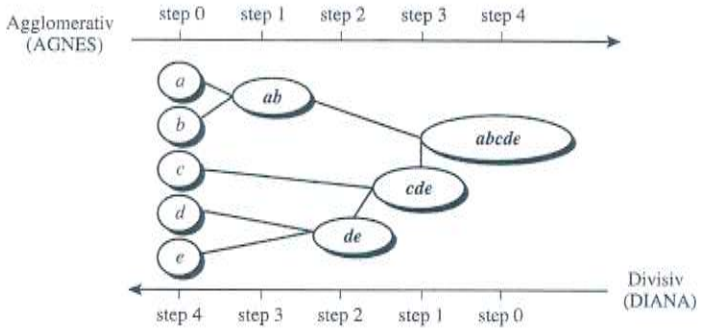

Abbildung 3.2: Agglomeratives und divisives Clustern. Entnommen: [HK06]

3.1.2.1 BIRCH

Der BIRCH-Algorithmus [ZRL96] ist zum Clustern großer Mengen numerischer Daten durch die Kombination von hierarchischem Clustering (im mikro-Clusteringschritt) und einer weiteren Clustering-Methode, wie beispielsweise iterativ partitionierendem Clustering (im makro-Clusteringschritt), entwickelt worden. Dabei bietet es eine Lösung für beide oben genannten Schwächen des hierarchischen Clusterings, indem es durch die Einführung so genannter Cluster-Feature-Bäume eine speichereffiziente Darstellung von Clustern wählt, welche dann in einem zweiten Cluste-

ringschritt noch einmal überarbeitet werden können, so dass Objekt-Verschiebungen zwischen Clustern möglich werden.

Cluster Features (CF) werden dabei als dreidimensionale Vektoren definiert, welche die Objektvektoren eines Clusters zusammenfassen. Seien n d-dimensionale Vektoren eines Clusters x_i gegeben, dann ist das CF für dieses Cluster

$$CF = \langle n, LS, SS \rangle , \tag{3.2}$$

wobei LS die lineare Summe (also $\sum_{i=0}^{n} x_i$) und SS die quadratische Summe (also $\sum_{i=0}^{n} x_i^2$) der n Objektvektoren sind. Cluster Feature sind damit eine Zusammenfassung über die Statistik eines Clusters und aus statistischer Sicht nichts anderes als das nullte, das erste und das zweite Moment eines Clusters. Da CF additiv sind und darüber hinaus ausreichen um jedes, in Kapitel 2.7 vorgestellte, Distanzmaß zu ermitteln, genügt es dem BIRCH-Algorithmus die Daten nur einmal einzulesen, um sie dann in einer CF-Baum-Struktur abzulegen. Alle weiteren Operationen können dann auf dem, im Speicher gehaltenen, Cluster-Feature-Baum durchgeführt werden, was die I/O Kosten erheblich verringert.

In der initialen Einlese-Phase werden die einzelnen Objektvektoren jeweils dem nächst liegendem Cluster bzw. dessen CF in dem balancierten CF-Baum zugeordnet. Dabei wird jeweils geprüft, ob das Threshold des Clusters, gemessen an Radius und Durchmesser, oder der Verzweigungsgrad des Baumknotens nach Hinzufügen des neuen Objekts überschritten wird. Sollte dies der Fall sein wird das Cluster in zwei neue Cluster aufgeteilt und der Baum umstrukturiert.

In einem zweiten Schritt wird eine weitere Clustering-Methode auf den Blättern des CF Baums angewendet, um zerstreute Cluster als Ausreißer zu entfernen und dicht aneinander liegende Cluster zu größeren Clustern zu fusionieren.

Experimente [ZRL96] konnten zeigen, dass BIRCH eine lineare Skalierbarkeit in Bezug zur Anzahl der zu gruppierenden Objekte bei gleichzeitig guter Qualität des Clusterings bietet. Da ein Knoten des CF-Baums aufgrund seiner Größe jedoch nur über eine begrenzte Aufnahmefähigkeit an Einträgen verfügt, kann es dazu kommen, dass ein CF-Baum-Knoten nicht unbedingt einem, dem Benutzer natürlich erscheinenden, Cluster entspricht. Darüber hinaus schneidet der BIRCH-Algorithmus nicht sonderlich gut ab, wenn das gesuchte Cluster nicht kugelförmig ist, da der Algorithmus Radius und Durchmesser als Threshold-Bedingung zur Aufteilung von Clustern verwendet.

Insofern, angepasst an die sessionweise Bündelung unserer Ausgangsdaten, erscheint BIRCH als eine der vielversprechensten Alternativen zu dem in dieser Arbeit vorgestellten Ansatz. Darüber hinaus wäre auch eine Kombination der beiden Ansätze interessant, da BIRCH durch die Cluster-Feature eine sehr effiziente Möglichkeit bietet, verschiedene Distanzmaße zu berechnen, die dann als Grundlage konfidenzbasierten Fusionierens verwendet werden könnten. Eine solche Kombination würde es eventuell sogar ermöglichen den von uns als offline Clearing Prozess entwickelten Ansatz online, sprich parallel zum laufenden Dialog, durchzuführen. Trotz dieser viel versprechenden Möglichkeiten, die BIRCH als Erweiterung bietet, soll hier noch einmal angemerkt werden, dass der Hauptfokus der hier vorgestellten Arbeit, die zuverlässige Detektion von Fehlern in der Wissensbasis und deren Bereinigung ist, während die Laufzeit bei einem problemlos offline durchführbaren Prozess weniger zentral erscheint.

3.1.2.2 ROCK

Der ROCK-Algorithmus [GRS99] basiert auf der Idee, Ähnlichkeit anhand von so genannten *Links* zwischen den zu untersuchenden Objekten zu bemessen. Dabei ist ein Link zwischen zwei Objekten durch die Anzahl gemeinsamer Nachbarobjekte definiert. Die beiden Objekte T_i und T_j gehören demnach zu dem gleichen Cluster, wenn sie viele gemeinsame Nachbarobjekte haben. Dies wird mit dem *Jaccard Koeffizienten* gemessen, der definiert ist als

$$sim(T_i, T_j) = \frac{|T_i \cap T_j|}{|T_i \cup T_j|}. \tag{3.3}$$

Liegt dieser Wert über einem zuvor definierten Schwellwert, werden die beiden Objekte in ein gemeinsames Cluster abgelegt. Dieser Ansatz wird vornehmlich für den Vergleich von Wahrheitswerten oder kategorischen Attributen genutzt.

Der ROCK-Algorithmus eignet sich damit hervorragend, um Ähnlichkeiten zwischen den im Dialog gesammelten Information ausfindig zu machen. Fehlertypen, wie der erste und der zweite Fehlertyp in Kapitel 2.4.1, bei denen für verschiedene Identitäten in der Wissensbasis dieselben Fragen ähnlich beantwortet wurden, ließen sich so eventuell gut auflösen. Dies kann jedoch nur gelingen, wenn für die jeweiligen Identitäten auch ausreichend Informationen im Dialog gesammelt wurden und die verwendeten Begriffe ähnlich genug sind, um solche Parallelen aufzudecken. Der dritte

Fehlertyp aus Kapitel 2.4.1 ließe sich mit ROCK überhaupt nicht auflösen. Damit wäre der ROCK-Algorithmus zwar keine Alternative, aber eine interessante Ergänzung, zu dem in dieser Arbeit vorgestellten Ansatz.

3.1.2.3 Chameleon

Der Chameleon-Algorithmus[KHK99] beruht auf der Beobachtung, dass Ansätze wie ROCK [GRS99] zwar die intra-Cluster-Distanz berücksichtigen, jedoch die inter-Cluster-Distanz gänzlich vernachlässigen, wohingegen andere Ansätze wie etwa CURE [GRS98] zwar die inter-Cluster-Distanz verwenden, dann jedoch die intra-Cluster-Distanz ignorieren. Chameleon begegnet dem Problem des Clusterings daher durch eine mehrteilige Architektur. Im ersten Schritt wird ein vollständiger k-nearest-neighbor graph aufgebaut, welcher im zweiten Schritt in kleine Teilgraphen zerlegt wird, die dann in einem letzten Schritt teilweise wieder zusammen gesetzt werden. Der Aufbau von Chameleon wird in Grafik 3.3 veranschaulicht.

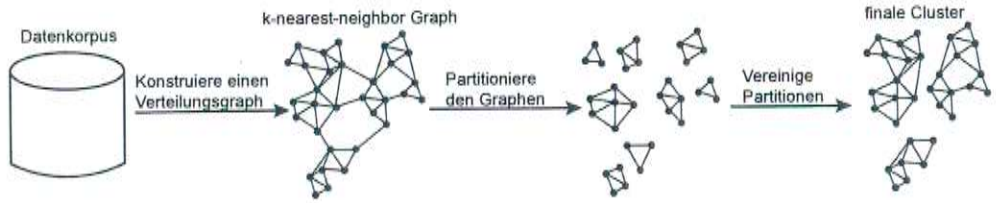

Abbildung 3.3: Chameleon: Hierarchisches Clustern auf Basis von k-nearest neighbors und dynamischer Modellierung. Entnommen und übersetzt aus: [KHK99]

Der Chameleon-Algorithmus ist insbesondere bei eigentümlichen Clusterformen, mit denen andere Cluster-Algorithmen wie BIRCH [ZRL96] oder DBSCAN [EKSX96] Schwierigkeiten haben, sehr stark. Wobei auf der anderen Seite seine Laufzeit von $O(n^2)$ bei n Objekten, inbesondere bei hoch dimensionalen Daten, zu einem Problem wird.

Würde man Chameleon, um die Sessionbündelung nicht aufzubrechen, statt auf den DCT-Bildvektoren auf den Centroiden der Sessions anwenden, wäre der Algorithmus bei unserem Szenario durchaus anwendbar. Allerdings hat die Tatsache, dass bei unseren Daten auf 100 Sessions (also hier initiale Centroide) etwa 28 Personen in der Groundtruth kommen, die Folge, dass die von uns gesuchten Zielcluster kaum mehr als 3 Sessions, bzw. aus Sicht des Chameleon-Algorithmus Datenpunkte, beinhalten.

Dieser Umstand lässt die Wahl des Chameleon-Algorithmus wenig sinnvoll erscheinen, würde man bei so kleinen Zielclustern doch gefahrlaufen, dass der erste und zweite Schritt des Algorithmus sich gegenseitig negieren.

In der vorliegenden Arbeit zeigen wir anhand des von uns vorgestellten Clustering-Verfahrens, wie sich auch ohne Mikro- und Makro-Clustering, inter- und intra-Clustering-Distanzen fürs Clustering vereinen lassen.

3.1.3 Dichtebasiertes Clustering

Die meisten Cluster-Verfahren beruhen auf Distanzmessungen zwischen den zu gruppierenden Objekten um Cluster auszumachen. Der Nachteil solcher Ansätze besteht darin, dass sie in der Regel kugelförmige Clusterformen begünstigen während andere Formen nur schwer erkannt werden. Eine Alternative bieten dichtebasierte Clusterverfahren. Die grundsätzliche Idee dabei ist ein Cluster so lange anwachsen zu lassen, bis dessen Dichte in der Umgebung einen gegebenen Schwellwert unterschreitet.

Da die von uns für das Clustering verwendeten Bilddaten ursprünglich für die FaceID verwendet wurden und man dort recht gute Ergebnisse mit distanzbasierten Verfahren wie k-nearest-neighbor erreicht [GHW08], lag der Verdacht nahe, dass auch für das Clustering ein distanzbasierter Ansatz genügen würde, was sich später in den Evaluationen bestätigte.

3.1.4 Modellbasiertes Clustering

Modellbasiertes Clustern geht von der Idee aus, dass sich jedes Cluster durch ein Modell beschreiben lässt, so dass es ausreicht anhand der Daten die Modellparameter zu schätzen, bzw. umgekehrt, anhand der Cluster-Modelle die Objekte auf Cluster zu verteilen. Beispielsweise könnte ein solches Modell die Cluster anhand der Dichteverteilungsfunktion über die Daten schätzen.

Da die von uns verwendeten DCT-Bildvektoren mit über 300 Merkmalen recht hochdimensional sind, was das Training guter Modelle vergleichsweise aufwendig macht, erschien ein modellbasierter Ansatz nicht naheliegend. Um die sessionweise Bündelung der Daten nicht aufzulösen wäre jedoch ein Ansatz, wie folgt denkbar gewesen: Zunächst trainiert man für die Daten jeder Session ein separates Modell, beispielsweise ein GMM[1], und verwendet anschließend eine Methode um die Ähnlichkeit zwischen den Modellen zu bestimmen. Sessions mit ähnlichen Modellen

[1] Gaussian mixture model

werden dann, wie in dem von uns vorgestellten Ansatz, iterativ fusioniert. Eine ähnliche Idee wird in [RQD00] verfolgt, um Sprecher auf Audiodaten zu erkennen.

3.1.5 Semi-supervised Clustering

Im Kontrast zu überwachten Lernverfahren, fehlt es dem Clustering an Anleitung durch den Benutzer in Form von gelabelten Klassen, mit der Folge, dass die erzielten Cluster nicht immer die gewünschte Form haben. Die Qualität des unüberwachten Clusterings lässt sich signifikant steigern, wenn man eine schwache Form von Überwachung durch den Benutzer gestattet. Dies kann beispielsweise in Form von Paarbeziehung zweier Instanzen, als dem gleichen Cluster oder verschiedenen Clustern zugehörig, geschehen. Eine solche auf Rückmeldung des Benutzers basierte Clustering-Methode bezeichnet man als *semi-supervised*. Letztlich handelt es sich bei Semi-supervised-Clustering, also um die Adaption des Active Learnings (siehe 3.2), auf das Problem des Clusterings.

Es lassen sich zwei unterschiedliche Methoden für Semi-supervised-Clustering unterscheiden: *Regelbasiertes semi-supervised Clustering* und *distanzbasiertes semi-supervised Clustering*. Die regelbasierten Ansätze beruhen auf, durch die Interaktion mit dem Benutzer erlernte, Regeln oder Labeln, um den Cluster-Algorithmus zu besseren Ergebnissen zu führen. Das bedeutet vor allem auch die Anpassung der Zielfunktion, basierend auf den erlernten Regeln, oder aber die Initialisierung und Reglementierung des Clustering-Prozesses auf Basis der gegebenen Label. Die distanzbasierten semi-supervised Clustering-Ansätze hingegen beruhen auf adaptiven Distanzmaßen, die auf Basis der erlernten Regeln oder Label optimiert werden. Ein adaptives Distanzmaß wäre dabei beispielsweise eine, durch den EM-Algorithmus[2] trainierte, Editier-Distanz.

In unserem Szenario ist das direkte Labeln von einzelnen Instanzen bzw. Bildern nicht möglich, da dies eine Wiedergabe der Bilder auf einem Bildschirm erfordern würde, was dem Konzept eines nur durch natürliche Sprache mit dem Benutzer interagierenden Robotors widersprechen würde. Somit wäre ein regelbasiertes semi-supervised Clustering nur als Ergänzung zu dem in dieser Arbeit vorgestellten Ansatz möglich. Als Ergänzung könnte man beispielsweise die, bei dem Korrektur-Dialog gesammelten, Informationen auch explizit für zukünftiges Clustering nutzen. Im derzeitigen Ansatz haben wir davon abgesehen, da wir die Fehlererkennung

[2]Erwartungswert Maximierung

und Datenbereinigung bewusst getrennt halten wollten. Eine weitere Möglichkeit, das semi-supervisede Clustering in den hier vorgestellten Ansatz mit einzubringen, wäre die Verwendung der bestehenden Label und einer darauf beruhenden adaptiven Distanz. Dies entspricht der von uns evaluierten Idee, die Editier-Distanz der Label als zusätzliches Merkmal für die Konfidenzberechnung mit einzubeziehen. Dies birgt jedoch, wie wir später sehen werden, das Problem, dass Fehler des dritten Typus (siehe Kapitel 2.4.1), statt aufgelöst, sogar begünstigt werden.

3.2 Active Learning

Ausgehend von der Beobachtung, dass überwachte Lernverfahren mit gut trainierten Modellen auch gute Klassfikationergebnisse bieten, für das Trainieren solcher Modelle jedoch große Mengen an gelabelten Daten benötigt werden, was in der Praxis sehr zeit- und kostenintensiv ist, versucht man beim *Active Learning* Verfahren anzubieten, die den Daten-Labelungsprozess möglichst effektiv gestalten. Die Idee ist, dass nicht alle Trainingsdaten gleich wichtig für das Training der Modelle sind, so dass man durch eine gezielte Auswahl an möglichst günstigen Referenzen die Modelle effektiver trainieren kann.

Klassischerweise geht man beim Active Learning wie folgt vor: Zunächst trainiert man die Modelle des Klassifikators mit einigen wenigen initial gelabelten Daten. Der so entstehende Klassifikator ist in seinen Entscheidungen noch sehr unsicher und die Erkennungsraten sind schwach. Im nächsten Schritt werden jene Instanzen aus den Trainingsdaten herausgegriffen, bei denen sich der Klassifikator am unsichersten ist, und der Benutzer wird gebeten diese zu labeln. Anschließend wird der Klassifikator mit den bis dahin gelabelten Daten erneut trainiert. Dieses Vorgehen wird solange iterativ wiederholt bis der Klassifikator die gewünschte Qualität erreicht.

Die verschiedenen Ansätze innerhalb des Active Learnings unterschieden sich vor allem darin, wie die zu erfragenden Instanzen ausgewählt werden. Hierfür lassen sich vor allem zwei grundsätzlich verschiedene Vorgehensweisen unterscheiden. Entweder wählt man Instanzen, die möglichst nah an den bestehenden Klassengrenzen liegen, oder aber jene, welche die größte Distanz zwischen zwei Klassen aufweisen. Beispiele für das erste Vorgehen findet man in [LG94], [TC01], [SC00] und [CCS00]. Wobei [LG94] jene ungelabelte Instanz als am unsichersten gewählt wird, deren bedingte Wahrscheinlichkeit am nächsten zu dem Wert 0.5 liegt, während

in [CCS00], [SC00] und [TC01] direkt mit der Distanz zu Klassengrenzen gearbeitet wird. Dieses an Klassengrenzen orientierte Vorgehen hat vor allem den Nachteil, dass die Verteilung der Daten nicht berücksichtigt wird. Das zweite Vorgehen findet insbesondere bei der Verwendung von SVMs[3] ihren Einsatz. Beispiele hierfür finden sich in [CCS00] und [TK01]. Eine dritte und klassifikatorunabhängige Möglichkeit Unsicherheit ausfindig zu machen, ist es, eine Gruppe von N Klassifikatoren über die einzelnen Instanzen entscheiden zu lassen und jene Instanzen als besonders fragwürdig auszuwählen, bei denen die größte Uneinigkeit zwischen den verschiedenen Klassifikatoren besteht. Beispiele für dieses Vorgehen finden sich in [FSST97] und in [SB02].

Eine Parallele zwischen Active Learning und der von uns vorgestellten Arbeit besteht insofern, als dass auch wir eine möglichst geschickte Auswahl an zu erfragenden Clustern bzw. Labeln für den Korrektur-Dialog wählen müssen. Allerdings können wir hierfür, im Gegensatz zu dem Vorgehen beim Active Learning, nicht auf die Ebene einzelner Instanzen bzw. Bilder zurückgreifen, da dies dem Entwurfsziel unseres Szenarios widerspricht. Ein weiterer bedeutsamer Unterschied besteht in unserem Ziel, denn im Gegensatz zu Active Learning-Ansätzen geht es bei unserer Arbeit nicht um die Verbesserung oder Optimierung eines Klassifikators, sondern um die Bereinigung einer Wissensbasis, deren Verunreinigung unabhängig[4] von der Qualität der ursprünglich auf den Daten klassifizierenden FaceID ist.

3.3 Duplikateliminierung auf Datenbanken

Das Problem der Duplikateliminierung, also das Erkennen und Beseitigen von Fehlern des ersten bzw. zweiten Typus (siehe Kapitel 2.4.1), wird in der Literatur für gewöhnlich im Kontext von Datenbanken behandelt. Duplikateliminierung ist in diesem Kontext ein Teilschritt des nötigen *Clearing* Prozesses, nachdem zwei oder mehrere heterogene Datenbanken zu einer neuen Datenbank zusammengefügt wurden. Denn bei der Datenfusion verschiedener Datenbanken ist nicht ausgeschlossen, dass mehrere, ursprünglich aus verschiedenen Datenbanken stammende Einträge, dieselbe Identität beschreiben.

[3] Support Vektore Machines

[4] Es gibt zwar durchaus einen Kausalzusammenhang, jedoch entstehen die eigentlichen Fehler, wie in Kapitel 2.4.1 beschrieben, primär im Dialog

Die Hauptschwierigkeit bei der Duplikateliminierung auf Datenbanken besteht folglich darin, diese Mehrfacheinträge ausfindig zu machen, obwohl die verwendeten Attribute und Kategorien aufgrund der heterogenen Natur der ursprünglichen Datenbanken sehr verschieden sein können.

Im Folgenden wird dieses Problem anhand einiger Beispiele aus der CiteSeer's Literatur-Referenz Datenbank veranschaulicht.[5] Aus unserem allgemeinen Weltwissen ist bekannt, dass zwei Referenzen Duplikate bilden, wenn Autor, Titel, Jahr und Publikationsort übereinstimmen. Allerdings kommen Referenzen in einer Vielzahl von Formaten vor, so dass es schwer ist, dieses Wissen in Regeln zu überführen. Als Beispiel dienen uns zwei Duplikate eines Buches, das auf CiteSeer in vierzehn verschiedenen Varianten eingetragen ist.

- L. Breiman, L. Friedman, and P. Stone, (1984). Classification and Regression. Wadsworth, Belmont, CA.

- Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charls J. Stone. Classification and Regression Trees. Wadsworth and Brooks/ Cole, 1984.

Anhand dieses Beispiels lässt sich leicht erkennen, dass Namen- oder Begriffsmatching für sich allein nicht ausreicht um Duplikate zu erkennen, denn einzelne Namen oder Begriffe können fehlen, falsch geschrieben oder abgekürzt vorliegen und auch die Reihenfolge kann sich verschieben. Auf der anderen Seite ist es, wie wir in den nächsten beiden Beispielen sehen werden, schwierig mit der Anzahl gemeinsamer Wörter zu arbeiten.

- H. Balakrishnan, S. Seshan, and R.H. Katz., Improving Reliable Transport and Handoff Performance in Celluar Wireless Networks, ACM Wireless Networks, 1(4), December 1995.

- H. Balakishnan, S. Seshan, E. Amir, R.H. Katz, „Improving TCP/IP over Wireless Networks," Proc. 1st ACM Conf. on Mobile Computing and Networking, November 1995.

Denn obwohl hier sehr viele gemeinsame Begriffe in beiden Referenzen vorkommen, handelt es sich in diesem Fall um kein Duplikat, im Gegensatz zum nächsten Beispiel, bei dem die beiden Duplikate deutlich weniger gemeinsame Worte teilen.

[5] Beispiele entnommen aus: [SB02]

- Johnson-Laird, Philip N. (1983). Mental models. Cambridge, Mass.: Harvard University Press.

- P. N. Johnson-Laird. Mental Models: Towards a Cognitve Science of Language, Inference, and Consciousness. Cambridge University Press, 1983.

Um dessen ungeachtet Duplikate auf großen Datenbanken effizient und automatisch - oder zumindest halbautomatisch - erkennen zu können, wurden in der Literatur einige Verfahren vorgeschlagen. Zwei der bekanntesten sollen im Folgenden vorgestellt werden.

3.3.1 Die Sorted-Neighborhood Methode

Ausgehend von zwei oder mehr Datenbanken, werden diese zunächst zu einer sequenziellen Liste der Größe N konkateniert und auf dieser Liste die *sorted-neighborhood* Methode angewendet. Dabei lässt sich die sorted-neighborhood Methode in drei Phasen einteilen. [HS95], [HS98]

1. **Erzeuge Schlüssel**: Berechne für jeden Eintrag der sortierten Liste auf Basis relevanter Merkmale einen Schlüssel.

2. **Sortiere Daten**: Sortiere die Liste in Abhängigkeit des in Schritt 1. erzeugten Schlüssels.

3. **Verschmelze**: Schiebe ein Fenster fester Größe über die sequenzielle Liste und vergleiche Einträge nur innerhalb des Suchfensters, um mögliche Duplikate zu finden.

Experimente konnten zeigen, dass das Verfahren bei mehrfacher Anwendung mit verschieden sortierten Listen und kleinen Suchfenstern eine höhere Erkennung von Duplikaten aufweist, als wenn einige wenige rechenaufwendige Durchläufe mit großen Suchfenstern berechnet werden.

Ohne Frage bildet eine der großen Stärken dieses Ansatzes die effiziente Nutzung von Datenbanken und ihren Möglichkeiten der Schlüsselgenerierung und Sortierung. Für die Anwendung auf numerischen DCT-Vektoren, deren einzelne Attribute nur wenig semantischen Inhalt tragen, erscheint der Ansatz jedoch nicht besonders naheliegend, selbst wenn wir das Problem der Sessionbündelung, das wir in unserem Szenario haben, außen vorlassen würden.

Davon abgesehen wäre die generelle Idee, eine wie auch immer geartete Sortierung der Cluster als Grundlage für die Duplikateliminierung zu nutzen, durchaus übertragbar. Man kann demnach durchaus Parallelen zwischen dem von uns verwendeten clusterbasierten Label-Mapping und der hier besprochenen sorted-neighborhood Methode ausmachen. Denn das von uns auf Sessions angewandte Clustering ließe sich in diesem Kontext auch als, nach einem bestimmten Schlüssel sortieren auffassen und das Mapping mit anschließendem Korrektur-Dialog als die Suche und Korrektur auf einem durch das Suchfenster bereitgestellten Ausschnitt der Daten.

3.3.2 Die Active Learning Methode

Möchte man das Problem der Duplikateliminierung mit Hilfe überwachter Lernverfahren lösen, so kommt den gelabelten Trainingsdaten eine entscheidende Rolle zu. Hierbei stellt sich vor allem das Problem eine breitgefächerte, repräsentative und herausfordernde Auswahl an Trainingsbeispielen in der Datenbank zu finden, die für das Training guter Modelle geeignet ist. Das Finden solcher Duplikate ist schwierig, denn dies erfordert vom Benutzer eine quadratische Suche auf den Einträgen der Datenbank. Noch schwieriger jedoch ist das Finden von interessanten Nicht-Duplikaten, die vom System mit großer Sicherheit fehlerhafterweise für Duplikate gehalten werden.

Um dem Problem der Trainingsdatenauswahl zu begegnen, schlagen Sarawagi und Bhamidipaty in [SB02] daher einen Active Learning Ansatz (vgl. Kapitel 3.2) vor, wobei mit Hilfe einiger weniger initial gelabelter Daten ein Klassifikator zur Erkennung von Duplikaten trainiert wird, der dann genutzt wird, um Beispiele auszuwählen, bei denen er sich am unsichersten ist. So wird der Klassifikator iterativ neu trainiert und somit verbessert und gleichzeitig genutzt, um die Trainingsdatenauswahl zu bestimmen.

4 Erkennung und Auflösung von Wissensbasisfehlern

In diesem Kapitel wird das in der vorliegenden Arbeit entwickelte Verfahren zur Erkennung und Auflösung von Wissensbasisfehlern erklärt. Dazu teilt sich das Kapitel in mehrere Teile auf. Im ersten Teil wird das Problem noch einmal kurz beschrieben und der gewählte Lösungsansatz anhand der Datenlage entwickelt. Im zweiten Teil wird das in dieser Arbeit entwickelte Clustering-Verfahren vorgestellt. Dieser Teil ist damit entscheidend für die Erkennung der Wissensbasisfehler. Im dritten und abschließenden Teil des Kapitels werden mögliche Verfahren zur Auflösung der Wissensbasisfehler präsentiert, wobei sowohl die durch das Clustering vorliegenden, als auch explizit durch einen Korrektur-Dialog gesammelte, Informationen genutzt werden.

4.1 Problembeschreibung und Datenlage

Wie in Kapitel 2.4 bereits geschildert gibt es eine Vielzahl an möglichen Fehlern, die sich im Laufe der Zeit in die Wissensbasis des IslEnquirers einschleichen können. Insbesondere die korrekte eins zu eins Beziehung zwischen der Wissensbasis-Identität samt des verwendeten Labels zu der real existierenden Person und ihrem Namen spielt dabei, wie in Kapitel 2.3 dargestellt, eine besonders wichtige Rolle. Werden diese Fehler nicht erkannt und bereinigt, kann dies die Qualität der Dialoge des IslEnquirers nachhaltig negativ beeinflussen. Personen, die mit dem System bereits gesprochen haben, können als solche nicht mehr wiedererkannt bzw. aufgelöst werden. Informationen verschiedener Personen werden untrennbar miteinander vermengt oder veraltete Informationen werden genutzt und ausgewertet. Dies alles gilt es zu verhindern.

Angesichts der Tatsache, dass es sich um eine überschaubare Anzahl von Sessions handelt, die einer Überarbeitung bedürfen, ließen sich diese Fehler natürlich leicht, durch eine direkte Korrektur der Datenbank bzw. der Wissensbasis seitens eines Experten, beseitigen. Das Szenario

des autonom lernenden IslEnquirers sieht einen solchen manuell Eingriff jedoch nicht vor. Somit verbleiben interne Prozesse und der Dialog zwischen System und Benutzer als Möglichkeiten die Wissensbasis zu bereinigen.

Ausgangspunkt unserer Überlegungen war es daher, wie man die innerhalb der IslEnquirer Dialoge gesammelten Informationen zur Lösung des Problems nutzbringend einsetzen könnte. Die Tatsache, dass der IslEnquirer sämtliche gesammelten Informationen sessionweise gebündelt abspeichert und, dass innerhalb einer Session davon ausgegangen werden kann, dass nur jeweils eine Person mit dem System interagiert, kommt uns dabei sehr entgegen, gewährleistet dies doch, dass Analysen dieser Daten auf der Session-Ebene möglich sind.

Daher ist es naheliegend Ähnlichkeiten zwischen den Daten verschiedener Sessions ausfindig zu machen und so eventuell fehlerhaft gelabelte Sessions aufzudecken. Auf diese Art und Weise lassen sich nicht nur die, in Kapitel 2.4.1 vorgestellten, Fehlertypen eins und zwei erkennen, sondern auch der dritte Fehlertyp. Der vierte Fehlertyp - im eigentlichen Sinne nicht falsch gelabelt - lässt sich so zwar nicht direkt auflösen, ist jedoch aufgrund seiner zeitlichen Korrelation vergleichsweise einfach durch einen Korrektur-Dialog aufdeckbar. In diesem Fall genügt es in einem Korrektur-Dialog[1] zu erfragen, ob eine Person, die mit dem System verdächtig lange nicht mehr interagiert hat, noch zur Groundtruth gehört. Folglich bedarf es zur Auflösung des vierten Fehlertypus - anders als bei den ersten drei Fehlertypen - keiner umfangreichen Analyse der Daten.

Wie bereits in Kapitel 2.4.1 angemerkt, entsteht ein Großteil der für diese Arbeit relevanten Fehler durch Fehlbestätigungen während des Dialogs. Damit ist nicht ausgeschlossen, dass auch andere Dialoginformationen fehlerhaft in der Wissensbasis vorliegen, so dass es fraglich erscheint, ob sich die Dialoginformationen für diesen Ansatz als Quelle der Ähnlichkeit verschiedener Sessions anbieten. Aber auch davon abgesehen, wäre ein Ähnlichkeitsvergleich sprachlicher Äußerungen, aufgrund der Ambiguität der Sprache, schwierig, denn bekanntlich lässt sich ein und derselbe Sachverhalt mit einer Vielzahl an möglichen Äußerungen und Begrifflichkeiten umschreiben. Damit scheidet die Sprachinformation als mögliche Quelle für einen Ähnlichkeitsvergleich verschiedener Sessions mehr oder minder aus, möchte man komplizierte ontologische und semantische Vergleiche vermeiden.

[1] siehe Kapitel 4.3.2

Neben den Dialog-Logdateien werden vom IslEnquirer auch Audio- und Videomitschnitte der Dialoge abgespeichert. Aus einer vorangegangenen Arbeit [GHW08] ist bekannt, dass sich die Audiodaten zum Zweck der Analyse nur wenig eignen, da aufgrund der Dialogführung ein Großteil der Mitschnitte aus kurzen Bestätigungsäußerungen (also „Ja"- oder „Nein"-Äußerungen) bestehen, die sich in ihrem personenspezifischen Klang nur wenig voneinander unterscheiden. Auf der anderen Seite wurden in der vorangegangenen Arbeit gute Ergebnisse mit der FaceID und den ihr zugrundeliegenden Videomitschnitten erzielt, so dass eine Analyse dieser Daten vielversprechend erschien.

4.2 Fehlererkennung durch Clustering auf Bilddaten

Die Videodaten liegen sessionweise gebündelt in Form von JPEGs der Auflösung 640×480 Pixel mit 24-bit Farbtiefe vor. Durch, die in Kapitel 2.5 beschriebenen, Vorverarbeitungsschritte lassen sich die einzelnen Bilder dann in 320 dimensionale Vektoren überführen. Genau genommen wird dabei nicht jedes Bild in einen solchen Vektor überführt, sondern nur jene, auf denen ein Viola & Jones Klassifikator [VJ03] ein Gesicht und zwei Augen erkannt hat.

Im Durchschnitt sind das auf unseren Testdaten etwa 400 Bilder pro Session, wobei diese Zahl je nach zeitlicher Länge der Session und der Kooperativität des Benutzers stark variieren kann. Einige Sessions beinhalteten deutlich über tausend DCT-Vektoren, eine einzelne Session sogar über zweitausend, während anderen Sessions nicht mal fünfzig DCT-Vektoren beinhalteten. Angesichts dieser Datenlage scheint es, neben den bisher genannten Argumenten, einmal mehr sinnvoll die sessionweise Bündelung der Daten für das Clustering nicht aufzubrechen. Andernfalls liefe man Gefahr die Datenverteilung als implizite Gewichtung ins Clustering miteinzubringen. Da Sessions jedoch unabhängig von der Anzahl aufgezeichneter Gesichtsbilder fehlgelabelt sein können, ist eine solche implizite Gewichtung nicht wünschenswert.

Um die im vorangegangenen Kapitel entwickelte Idee einer Ähnlichkeitsanalyse auf Session-Ebene nun auf Basis der Video- bzw. Bilddaten zu realisieren scheint Clustering eine sinnvolle Wahl. Bietet uns das Clustering doch die Möglichkeit Muster auf Daten zu erkennen, ohne zuvor aufwendig einen Klassifikator trainieren zu müssen. Insbesondere die Tatsache,

dass keine gelabelten Daten für das Training benötigt werden, macht diesen Ansatz für unser Szenario eines autonom lernenden Dialog Robotors praktikabel.

4.2.1 Cluster-Distanzen

Der naheliegenste Ansatz Clustering auf Bildvektoren zu realisieren, ohne dabei die sessionweise Bündelung der Daten aufzubrechen, ist statt direkt auf den Bildvektoren zu clustern, jeweils sessionweise Repräsentanten zu verwenden. Eine einfache Möglichkeit dies zu tun, ist für jede Session auf Basis der in ihr beinhalteten DCT-Vektoren einen Durchschnittsvektor - auch *Centroide* genannt - zu berechnen und diesen für alle weiteren Clusteringschritte zu verwenden. Dies birgt vor allem den Vorteil auf sehr einfache Art und Weise die Distanz zwischen verschiedenen Clustern - auch inter-Cluster-Distanz genannt - berechnen zu können, indem man auf die Euklid- oder wahlweise auch auf die Manhattan-Distanz zwischen den Centroiden zurück greift.

Die *inter-Cluster-Distanz* ihrerseits ist einer der zentralen Werkzeuge, möchte man distanzbasiert Ähnlichkeiten zwischen Cluster ausfindig machen. Definieren wir eine Session bzw. die ihr zugehörigen Bildvektoren jeweils als ein Cluster, entspricht die inter-Cluster-Distanz somit der inter-Session Distanz. Räumlich nah aneinander liegende Cluster bedeuten dann also folglich ähnliche Sessions, oder, um sprachlich präzise zu sein, ähnliche Gesichter beschreibende Bildvektoren innerhalb der Sessions bzw. Cluster.

Ein weiterer wichtiger Begriff in diesem Zusammenhang ist die *intra-Cluster-Distanz*. Sie drückt die Distanz innerhalb eines Clusters aus und beschreibt so in gewisser Weise die Streuung oder Varianz der Vektoren eines Clusters.

Da die hier vorgestellte Arbeit letztlich darauf hinausläuft, Personen, ungeachtet des im Dialog erlernten Labels, auf den Bilddaten wiederzuerkennen bzw. verschiedene Personen trotz gleichem Label unterscheiden zu können, spielt ein dritter Begriff eine wichtige Rolle: Die *inter-Personen-Distanz*. Alle drei Begriffe werden in Abbildung 4.1 schematisch veranschaulicht.

Eine wichtige Voraussetzung, dass der Clustering-Ansatz zur Erkennung von Wissensbasis Fehlern funktioniert, ist, dass die inter-Personen-Distanz auf den verwendeten Daten größer ist, als die inter-Session-Distanz bei einer Person (intra-Personen-Distanz). Ist dies nicht der Fall, lassen sich Personen schlicht und ergreifend nicht anhand der Cluster-Di-

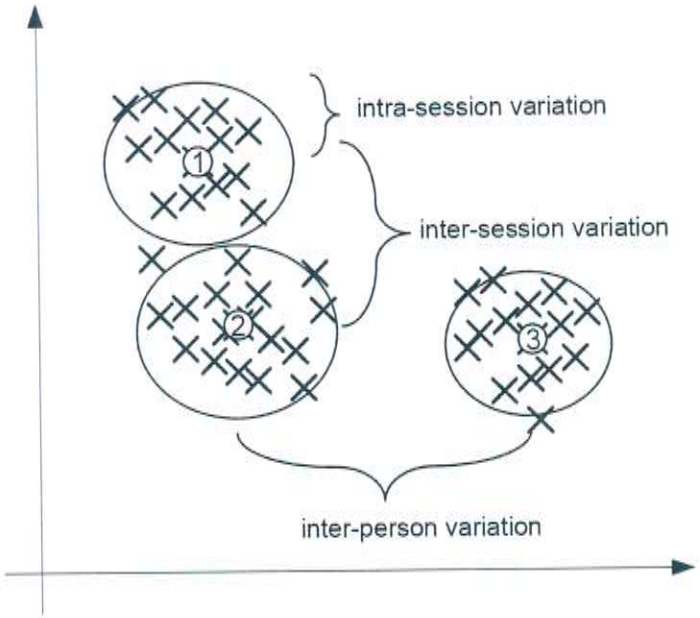

Abbildung 4.1: Bildliche Darstellung der intra-Cluster-Varianz, inter-Cluster-Varianz und der inter-Personen-Varianz. Entnommen: [Hol09]

stanzen unterscheiden. Einen bedeutsamen Einfluss darauf hat auch die intra-Session-Distanz. Denn sind die intra-Session-Distanzen generell größer als die inter-Session-Distanzen, besteht die Gefahr, dass sich die Sessions derart überlagern, dass keine personenspezifischen Muster (Zielcluster) in den Daten zu erkennen sind. Im ersten Fall, wenn also die inter-Personen-Distanz nicht größer ist als intra-Personen-Distanz, lässt sich im Zweifelsfall durch eine geschickte Wahl des Distanzmaßes und einer eventuellen Dimensionsreduktion Abhilfe schaffen. Im zweiten Fall, wenn keine personenspezifischen Muster in den Daten existieren, hingegen nicht. Die Tatsache, dass die Daten bereits erfolgreich für eine distanzbasierte FaceID Klassifikation [GHW08] genutzt wurden, legt jedoch nahe das Daten personenspezifische Muster enthalten.

Trotzdem macht es durchaus Sinn, neben der zu Beginn des Kapitel erwähnten Distanzmethode, noch weitere Distanzmaße in Erwägung zu ziehen. Schließlich handelt es sich bei einem centroidbasierten Distanzmaß letztlich um eine Heuristik, die wie jede Heuristik auch ihre Schwächen hat.

Wir haben in der vorliegenden Arbeit daher vier Distanzmaße untersucht:

1. Eine centroidbasierte Euklid-Distanz
2. Die durchschnittliche inter-Cluster-Distanz
3. Die durchschnittliche Editier-Distanz
4. Eine gewichtete Summe aus 1. und 3.

Die ersten zwei Distanzmaße wurden bereits in Kapitel 2.7 vorgestellt, so dass wir uns hier auf die letzten beiden konzentrieren wollen. Es sei jedoch noch einmal angemerkt, dass die inter-Cluster-Distanz einen Aufwand von $O(n \cdot m)$ hat, wobei n und m für die Anzahl an Bildvektoren in den zu vergleichenden Clustern steht. Somit ist dieses Distanzmaß wesentlich rechenaufwendiger als ein centroidbasierter Ansatz und in der Regel nur zu rechtfertigen, wenn letzterer wesentlich schlechtere Ergebnisse liefern sollte.

Die Editier-Distanz - auch als Levenshtein-Distanz[2] bezeichnet - ist ein informationstheorisches Maß für den Unterschied zwischen zwei Zeichenketten. Sie drückt dabei die minimale Anzahl der Operationen (Einfügen, Löschen und Ersetzen) aus, die nötig sind, um eine Zeichenkette in die andere zu überführen. In unserem Fall interessiert uns die Editier-Distanz zwischen den Labeln der Sessions, weisst eine kleine Editier-Distanz doch auf mögliche Buchstabier- bzw. Rechtschreibfehler hin. Da ein Cluster jedoch, wie wir in Kürze sehen werden, durchaus auch mehr als eine Session beinhalten kann und somit auch mehr als nur ein Label, musste das Distanzmaß dahingehend angepasst werden. Zu diesem Zweck werden die Session-Label an die dazugehörigen Bildvektoren vererbt und statt die Editier-Distanz direkt zwischen Clustern zu bestimmen, wird sie als durchschnittliche Editier-Distanz aller Vektoren, der zu vergleichenden Cluster, ermittelt. Die durchschnittliche Editier-Distanz ist damit definiert als:

$$D_6 = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} E(\vec{X}_i, \vec{X}_j)}{n \cdot m} \tag{4.1}$$

wobei $E(\vec{X}_i, \vec{X}_j)$ für die Editier-Distanz zwischen den zu vergleichenden Label, der Vektoren $\vec{X}_i$ und $\vec{X}_j$ bzw. der entsprechenden Sessions,

[2] nach dem russischen Wissenschaftler Wladimir Lewenstein bezeichnet, der sie 1965 einführte.

steht. Während n und m jeweils für die Anzahl an Bildvektoren in den zu vergleichenden Clustern stehen.

Ist in den zu vergleichenden Clustern nur jeweils ein Label vertreten, entspricht die der klassischen Editier-Distanz der beiden Label. Andernfalls entspricht sie einer anteiligen Gewichtung der Label in den Clustern. Somit bringt die durchschnittliche Editier-Distanz auch eine implizite Gewichtung der eventuell unterschiedlich großen Sessions innerhalb eines Clusters mit ein, was im allgemeinen zwar nicht ideal, hier aber im Kontext der Cluster-Zusammensetzung durchaus gewollt ist.

Um sowohl Label-Ähnlichkeit also auch die Bildvektor-Ähnlichkeit nutzen zu können, haben wir noch ein weiteres hybrides Distanzmaß eingeführt. Es berechnet sich aus einer gewichteten Summe der Euklid-Distanz und der Editier-Distanz. Oder, um es in einer Formel auszudrücken:

$$D_7 = \alpha \cdot D_2 + \beta \cdot D_6, \tag{4.2}$$

wobei α und β die Gewichte ausdrücken, während D_1 für die, in Gleichung 2.2 definiert, Euklid-Distanz und D_6 für die, in Gleichung 4.1 definierte, durchschnittliche Editier-Distanz stehen.

4.2.2 Iteratives Clustering

Bis zu diesem Punkt haben wir davon gesprochen, dass es sinnvoll ist die Bilddaten einer Session gebündelt als ein Cluster zu verstehen und als solches zu initialisieren. Weiter haben wir davon gesprochen, dass man anhand der Distanz zwischen den Clustern Ähnlichkeiten zwischen den Sessions ausfindig machen kann.

Ziel ist es jedoch nicht nur einzelne Beziehungen zwischen den Sessions zu erfassen, sondern idealerweise sämtlich fehlgelabelte Sessions als solche zu erkennen. Aus Clustering-Perspektive bedeutet dies, dass unsere Zielcluster der Fusion aller Sessions einer Person entsprechen. Oder um es nochmal anders auszudrücken: Wir würden gerne so clustern, dass am Ende des Clusterings die Anzahl der Cluster der Anzahl an Personen in der Wissensbasis entspricht und jedes Cluster nur Daten genau einer Person enthält. Aus diesem Grund ist es unumgänglich unsere initialen Cluster, die jeweils nur eine Session enthalten haben, schrittweise zu neuen Clustern zu fusionieren, die dann zwei oder mehr Sessions beinhalten. Wir verwenden folglich einen agglomerativen Clustering Ansatz, der Cluster anhand ihrer Distanz zueinander iterativ fusioniert. Abbildung 4.2 ver-

anschaulicht dieses Vorgehen schematisch anhand eines centroidbasierten Distanzmaßes.

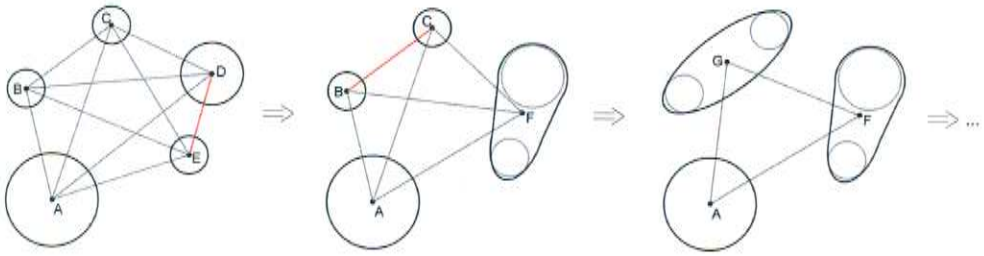

Abbildung 4.2: Schematische Darstellung eines centroidbasierten iterativ agglomerativen Clusterings

Die Kreise stehen dabei für die Datenverteilung innerhalb der Cluster, während die Punkte die Centroide darstellen. Wie man sieht wird der Distanzgraph zwischen den Centroiden aufgespannt, wobei die jeweils kürzeste Distanz rot hervorgehoben wurde. Schrittweise werden die jeweils nächsten Cluster zu neuen Clustern fusioniert, was bedeutet, dass diese neuen Cluster auch neue Centroiden haben, was wiederum zur Folge hat, dass sich auch die Distanzen zu den anderen Clustern mit jedem Iterationsschritt verschieben.

So wird klar, dass bei einer iterativen Wiederholung dieses Vorgehen der Großteil des Rechenaufwands durch die stetige Neubestimmung der Distanzen generiert wird. Dies ist ein klares Argument gegen ein rechenaufwendiges Verfahren, wie die durchschnittliche inter-Cluster-Distanz, und für einen heuristischen Ansatz, wie es das centroidbasierte Vorgehen ist. In jedem Fall ist der Rechenaufwand jedoch direkt proportional zur Anzahl der betroffenen Bildvektoren, sei es nun durch die Distanzbestimmung, wie im ersten Fall, oder durch die Neubestimmung der Centroiden, im zweiten Fall.

Um den Aufwand zumindest etwas einzuschränken haben wir uns daher entschlossen die Anzahl der verwendeten Bildvektoren pro Session zu beschränken. Insbesondere bei Sessions mit weit überdurchschnittlich vielen Bildvektoren stellt sich die Frage, ob das mehr an Bildern im Clustering-Ergebnis dem mehr an Rechenaufwand gegenübersteht. Zumindest aus dem Bereich der Gesichtserkennung auf DCT-Vektoren ist bekannt, dass 200-300 Bilder pro Person in der Regel absolut ausreichend sind, um gute Erkennungsraten zu erzielen. Wir haben die Anzahl an verwendeten Bild-

vektoren deshalb für jede Session auf 400 beschränkt, wobei die Auswahl dem Zufall überlassen wurde. Darüberhinaus führen wir für jedes Cluster eine Outlier-Detektion durch. Die initialen Session-Cluster werden um 20% der Bildvektoren bereinigt, die am weitesten vom Cluster-Centroiden entfernten sind, und die später fusionierten Cluster noch einmal um jeweils 10%. Das Filtern von Ausreißern soll gewährleisten, dass die verwendeten Bildvektoren repräsentativ für ihr Cluster sind. So wird durch das initiale Filtern unter anderem sichergestellt, dass Personen die möglicherweise im Hintergrund der Video-Aufnahme mit aufgezeichnet wurden und so eventuell ebenfalls wie der Gesprächspartner mit DCT-Vektoren in der Session vertreten sind, das Clustering nicht verfälschen; Während das Outlier-Filtern auf den fusionierten Clustern mehr das Ziel verfolgt die intra-Cluster-Distanz niedrig zu halten, um die Überschneidungen der iterativ anwachsenden Cluster möglichst klein zu halten.

4.2.3 Abbruchkriterium & Adaptive Distanz

Das Clustering-Verfahren, so wie wir es bis jetzt beschrieben haben, würde die Session-Cluster solange iterativ fusionieren, bis schließlich nur noch ein großes, alle Session und Personen auffassendes, Cluster verbleiben würde. Eine Abbruchbedingung für das iterative Clustering ist somit von zentraler Bedeutung. Bei dem von uns verfolgten Ziel fehlgelabelte Sessions anhand der zusammengeclusterten Sessions zu erkennen, muss der Clustering-Prozess folglich abgebrochen werden, bevor Cluster entstehen, die Daten unterschiedlicher Personen enthalten. Denn in diesem Fall würden Labelkonflikte innerhalb des Clusters nicht auf fehlgelabelte Sessions hinweisen, sondern nur ein Fehlclustering beschreiben.

Daher arbeiten wir mit der, in Kaptiel 2.6 vorgestellten, logistischen Regression, um auf Basis verschiedener Merkmale, wie etwa der Distanz zweier Cluster, eine Aussage darüber zu treffen, ob eine gegebene Clusterfusion korrekt ist oder nicht. Wir bestimmen also eine Konfidenz über die Richtigkeit einer Clusterfusion. Diese Konfidenz wird durch eine Wahrscheinlichkeit ausgedrückt. Genauer gesagt der Wahrscheinlichkeit, dass die zwei zu fusionierenden Cluster, nur Bildvektoren bzw. Sessions der gleichen Person enthalten und folglich in ein gemeinsames Zielcluster gehören. Als Abbruchbedingung wird ein Schwellwert für die Konfidenz festgelegt, der nicht unterschritten werden darf. Der Schwellwert entspricht so der zugelassenen Fehlerquote des Clustering-Prozesses, und kann entsprechend angepasst werden. Ein hoher Schwellwert garantiert eine geringe Fehlfusionsrate, dies aber bedeutet gleichzeitig, dass der iterative Cluste-

ring-Prozess früher beendet wird, und demnach insgesamt weniger Cluster fusioniert werden, was im Zweifelsfall zur Folge hat, dass weniger fehlerhafte Label aufgelöst werden können. Ein zu niedriger Schwellwert hingegen unterbricht den iterativen Clustering-Prozess zu spät, mit der Folge, dass Cluster entstehen, die Daten von mehr als nur einer Person umfassen, was wiederum der korrekten Auflösung fehlerhaft gelabelter Sessions entgegensteht.

Als Merkmal für die Konfidenzbestimmung eignet sich, unabhängig von dem letztlich verwendeten Distanzmaß, die Distanz zwischen den zu fusionierenden Clustern, wobei die Distanz nicht nur absolut, sondern auch relativ zu den anderen Distanzen im Distanzgraph angegeben werden kann. Die relative Distanz (relDist) wird dabei wie folgt aus der absoluten Distanz (absDist) abgeleitet:

$$\text{relDist} = \frac{\text{absDist}}{max(\text{absDist})}. \tag{4.3}$$

Dies birgt den Vorteil einer etwas größeren Robustheit, denn die absolut angegebenen Distanzwerte werden stärker durch die Verteilung auf dem Datenkorpus beeinflusst, als relative. Und da uns für die Fusion eigentlich nur interessiert welche Cluster am nächsten zueinander stehen und weniger mit welchem exakten Distanzwert, scheint die Verwendung einer relativen Distanzangabe eigentlich geschickter. Auf der anderen Seite wird eine relative Distanzangabe während des iterativen Clusterings stärker beeinflusst als absolut angegebene Distanzwerte. Absolute Distanzwerte bieten sich somit mehr für die Verwendung fester Schwellwerte an, ab deren Überschreitung davon ausgegangen werden kann, dass Cluster verschiedener Personen fusioniert würden. Bei der Wahl zwischen absolut oder relativ angegebenen Distanzen steht demnach die Generalisierbarkeit, der von uns bestimmten Logit-Koeffizienten auf anders verteilte Datenkorpora, der exakten Bestimmung des bestmöglichen Abbruchpunkts, für den iterativen Clustering-Prozess, gegenüber. Da die Abwägung zwischen diesen beiden Punkten schwierig erscheint, haben wir die logistische Regression für beide Distanzdarstellungen evaluiert.

Neben der inter-Cluster-Distanz eignet sich noch ein weiteres Merkmal für die Konfidenzbestimmung, nämlich die intra-Cluster-Distanz, oder Clusterdichte. Denn je größer ein Cluster im Lauf des iterativen Fusionierens wird, desto unspezifischer wird es auch. Am anschaulichsten wird dies anhand der Centroide. Hat der Algorithmus erst einmal damit an-

gefangen Cluster verschiedener Personen zu fusionieren, so entspricht der Durchschnittsvektor eines solchen personenübergreifenden Clusters weder Person A noch Person B, sondern vielmehr einem Durchschnittsgesicht aus A und B. Dieses Durchschnittsgesicht hat dann allerding gute Chancen näher an einem Cluster der Person C zu liegen, als irgend eines der anderen Cluster der Person C. Das liegt unter anderem daran, dass Aufnahmebedingungen, wie etwa Beleuchtungsverhältnisse, in einem nicht zu unterschätzenden Ausmaß in die DCT-Vektor Berechnung miteinfließen, mit der Folge, dass wir stets Gefahr laufen statt Cluster verschiedener Personen Cluster verschiedener Beleuchtungsverhältnisse zu erzeugen [MAU94]. Die intra-Cluster-Distanz als Merkmal für die Konfidenzbestimmung hilft uns daher Clusterfusionen zunächst zurückzustellen, selbst wenn die Cluster von der inter-Cluster-Distanz her räumlich am nächsten zueinander stehen. Da die durchschnittliche intra-Cluster-Distanz, insbesondere bei immer größer werdenden Clustern, recht rechenintensiv ist, approximieren wir diese durch die Anzahl beinhalteter Sessions in einem Cluster. So stellen wir Fusionen größerer Cluster tendenziell zurück, was im Zweifelsfall, je nach Konfidenzschwellwert, bedeutet, dass nicht alle Sessions einer Person in einem finalen Cluster vereinigt werden. Das ist jedoch insofern vertretbar, da es genügt, wenn eine fehlgelabelte Session mit einer anderen Session der gleichen Person zusammen geclustert wurde, um einen Labelkonflikt als solchen zu erkennen.

Bis hier haben wir davon gesprochen, dass wir Cluster anhand ihrer Distanz zueinander iterativ fusionieren und anhand einer Konfidenz den iterativen Prozess abbrechen. Viel mehr Sinn macht es jedoch die Konfidenz selbst als Ähnlichkeitsmaß zwischen den Clustern zu verstehen. Der Distanzgraph wird somit nicht explizit durch die Distanzen zwischen den Clustern aufgespannt, sondern implizit mittels der Konfidenzen, gewissenmaßen in Form eines Konfidenzgraphen. Die Konfidenz stellt in dem Sinne folglich ein adaptives Distanzmaß dar, wie wir es in Kapitel 3.1.5 angesprochen haben.

4.3 Auflösung der fehlerhaften Session-Label

Nach Ablauf des in Kapitel 4.2 beschriebenen Clustering-Verfahren liegen die Session gebündelt in Cluster vor, die hoffentlich nicht mehr als eine Person umfassen. Sind nur einige der in den Clustern gebündelten Session fehlgelabelt, treten innerhalb der Cluster zwangsläufig Labelkonflike

auf. Will heißen, dass ausgehend von verschiedenen Sessions verschiedene Label in einem Cluster vereint vorliegen. Dies kann entweder auftreten, wenn es während des Clusterings Fehlfusionen gab, oder aber, wenn mindestens eines der im Konflikt zu einander stehenden Label falsch ist. Im folgenden Kapitel wollen wir uns damit befassen, wie man diese Labelkonflikte auflösen kann, ohne zu wissen, welches der Label der Groundtruth entspricht.

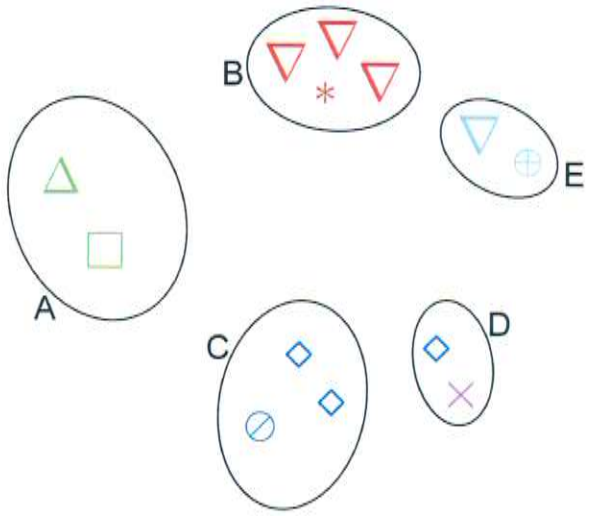

Abbildung 4.3: Schematische Darstellung der finalen Cluster und der in ihnen vertretenen Session-Label, mit Groundtruth-Wissen

In Abbildung 4.3 wird eine schematische Darstellung einer möglichen Clusteraufteilung nach Ablauf des Clustering-Verfahren gezeigt. Die Kreise stehen, wie zuvor, für die Cluster, während die verschiedenen Symbole für verschiedene Label und die Farben für verschiedene Personen stehen. Wie man leicht anhand der Farbverteilungen innerhalb der Cluster erkennen kann, entsprechen die Cluster A, B, C und E genau der Clusteraufteilung, wie wir sie uns durch das Clustering erhoffen. Bei dem Cluster D hingegen handelt es sich offensichtlich um ein Beispiel eines durch Fehlfusion entstandenen Cluster. Mit dem Wissen um die Groundtruth, hier durch die Farbkodierung gegeben, erkennt man leicht, dass es sich bei ⊘ und ∗ offensichtlich um Fehler des ersten bzw. zweiten Typus handelt. Und angesichts der Tatsache dass ▽ in Cluster B so häufig bestätigt wurde, handelt es sich bei dem Label ▽ in Cluster E offensichtlich um einen typischen Fehler des dritten Typus.

All das erscheint mit dem Wissen um die Groundtruth bzw. dank der Farbkodierung denkbar einfach. Doch unglücklicherweise steht uns dieses Wissen für die Auflösung der Labelkonflikte nicht zur Verfügung und so ähnelt unsere Ausgangssituation viel mehr der Darstellung in Abbildung 4.4. Dort erscheint keine der Zuordnungen klar und jedes der Cluster könnte möglicherweise auch durch eine Fehlfusion entstanden sein. Hier wird dann offensichtlich, dass selbst wenn unser Clustering-Verfahren die Daten absolut perfekt in Cluster aufteilen sollte, die korrekte Auflösung fehlerhafter Labels ein nicht zu unterschätzendes Problem bleibt.

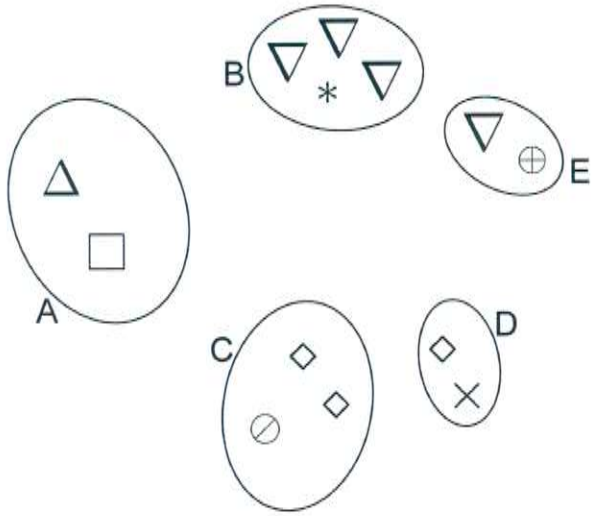

Abbildung 4.4: Schematische Darstellung der finalen Cluster und der in ihnen vertretenen Session-Label, ohne Groundtruth-Wissen

Die sicherste Methode, um eine fehlerfreie Wissensbasis zu garantieren, ist es alle Sessions aus der Wissensbasis zu streichen, die in einem Cluster aufgeführt sind, mit ein oder mehr Labelkonflikten. Denn jede dieser Sessions könnte möglichweise fehlgelabelt sein oder aufgrund eines fehlerhaften Labelmappings verfälscht werden.

Verständlicherweise ist ein solch restriktives Vorgehen nicht immer angemessen und so haben wir alternative Methoden untersucht, um, auf Basis der uns zur Verfügung stehenden Clusterinformationen, *Label-Mappings* abzuleiten. Ein Label"=Mapping ist in diesem Zusammenhang nichts anderes als das Ersetzen einiger Session-Label. Diese Veränderung kann, muss aber nicht, dauerhaft in der Wissensbasis geschehen. Eine separate Mapping-Tabelle bietet beispielsweise die Möglichkeit die Veränderung

über mehrere Clustering-Durchläufe hinweg zu beobachten. Außerdem bietet eine Mapping-Tabelle im Gegensatz zu einem direkten Eingriff in die Wissensbasis die Möglichkeit je nach Anwendung unterschiedlich vorzugehen. So können fragliche Sessions restriktiv aus dem Korpus genommen werden, entsprechend der Mapping-Tabelle gemappt oder aber gänzlich unverändert übernommen werden. So wäre beispielsweise für die vom IslEnquirer verwaltete Internetseite das Verwenden der gemappten Label sinnvoll, wohingegen man für das Training der FaceID möglicherweise ein restriktiveres Vorgehen bevorzugt.

4.3.1 Clusterbasiertes Mapping

Der einfachste und direkteste Weg auf Basis der Clusterinformationen ein Session-Label-Mapping abzuleiten, ist es die Labelverteilung innerhalb der Cluster zu verwenden. Ein solches Vorgehen ist insofern möglich, weil es recht wahrscheinlich ist, dass korrekte Labels in mehreren Session wiederholt bestätigt werden, wohingegen fehlerhafte Labels, eher selten ein weiteres Mal bestätigt werden. Zumindest Fehler des zweiten und dritten Typus (siehe 2.4.1) können auf diese Weise, sofern korrekt geclustert, recht zuverlässig aufgelöst werden. Und bei Fehlern des ersten Typus, mag zwar nicht zwingend das Label mit der korrekten Schreibweise gewählt werden, aber zumindest wird das Label anschließend einheitlich verwendet, so dass man ebenso von einer korrekten Auflösung sprechen könnte.

Um die Labelverteilung innerhalb der Cluster fürs Label Mapping zu nutzen, übergibt man die Verteilung der Label als Hypothesen-Liste[3] mit entsprechenden Wahrscheinlichkeiten an die einzelnen Sessions. Für das Cluster B in Abbildung 4.4 beispielsweise würde die Verteilung $\frac{3}{4}\nabla$ zu $\frac{1}{4}*$ dazu führen, dass sämtliche Sessions innerhalb des Clusters eine Label-Liste der Art 75% ∇, 25% $*$ erhielten und folglich auf das Label ∇ gemappt würden, was, wie man anhand von Grafik 4.3 sehen kann, richtig wäre. So hätten wir in diesem Fall das Label $*$ korrekt aufgelöst.

In den später folgenden Evaluationen in Kapitel 5.4 wird sich zeigen, dass dieses Mapping-Verfahren erstaunlich gute Ergebnisse liefert. Dennoch weist es auch erhebliche Mängel auf. Wie jedes auf Clusterinformationen basierte Mapping-Verfahren ist es zunächst recht anfällig für Fehlfusionen, welche jedoch bei einer geschickten Wahl des Konfidenzschwellwerts (siehe 4.2.3) weitgehend vermeidbar sind. Vielmehr allerdings hat das Verfahren Schwächen bei einem viel häufiger zu beobachtenden Fall,

[3] später als Clustered Session List (CSL) bezeichnet

wenn nämlich die finalen Cluster nur wenige Sessions enthalten und der Labelkonflikt folglich einem 50-50 Verhältnis entspricht, wie dies etwa in Abbildung 4.4 bei den Clustern A, D und E der Fall ist. Die korrekte Auflösung ist dann einzig der alphabetischen Reihenfolge der Label bzw. dem Zufall geschuldet.

4.3.2 Korrektur-Dialoge

Da wir im vorangegangenen Kapitel gezeigt haben, dass die Labelverteilung innerhalb der Cluster nicht immer ausreicht, um eine korrekte Session-Label-Auflösung zu garantieren, ist es sinnvoll weitere Informationsquellen zu nutzen. Angesichts der Tatsache, dass wir mit dem Clustering die Möglichkeiten der Datenanalyse weitestgehend ausgeschöpft haben, liegt es nahe die Dialogfähigkeit des IslEnquirers zu nutzen. Daher sehen wir vor, dem IslEnquirer die Möglichkeit zu bieten, in einem Korrektur-Dialog sich mit einer *trusted-Person* über Namen bzw. Label auszutauschen, die ihm nach dem Clustering fragwürdig erscheinen. Eine trusted-Person ist in dem Fall ein Administrator oder eine andere vertrauenswürdige Person, bei der angenommen werden kann, dass sie die Personen bzw. Namen in der Groundtruth kennt und die Fragen des IslEnquirers wahrheitsgemäß beantwortet.

Da die Zeit, die eine trusted-Person für einen solchen Dialog aufwendet, eine kostbare Ressource ist, sollte eine Voranalyse der Daten, ähnlich dem Vorgehen beim Active Learning, genutzt werden, um zu erfragende Label nach Priorität zu sortieren. Die von uns verwendete Methode zur Sortierung der Label wird in Kapitel 4.3.2.1 vorgestellt und mündet in einer so genannten „sortierten" *Clustered-Label-List* (CLL). In Abgrenzung dazu wird die Information über die Label-Verteilung innerhalb der Cluster, wie wir sie in Kapitel 4.3.1 eingeführt haben, als *Clustered Session List* (CSL) bezeichnet. Zunächst jedoch soll im Folgenden der generelle Aufbau möglicher Korrektur-Dialoge dargestellt werden. Abbildung 4.5 zeigt den von uns vorgesehenen Aufbau dreier separater Korrektur-Dialoge.

Der erste Korrektur-Dialog, auch als Existenz-Dialog bezeichnet, erfragt bei der trusted-Person die Existenz der Label bzw. ob eine Person mit diesem Namen bekannt ist. Dieser Korrektur-Dialog eignet sich insbesondere zur Auflösung von Fehlern des zweiten Typus, kann aber auch, wie wir später sehen werden, zur Erkennung der anderen Fehlertypen beitragen.

Der zweite Korrektur-Dialog, bereits in Kapitel 4.1 motiviert, ist primär zur Auflösung von Fehlern des vierten Typus gedacht. Hier werden bereits bestätigte Label nach einer gewissen Zeit noch einmal überprüft, wenn

der IslEnquirer die entsprechende Person über einen längeren Zeitraum nicht mehr gesehen bzw. gesprochen hat. Label bzw. Sessions mit Labeln auf der Blacklist werden aus dem Korpus entfernt und folglich in späteren Clustering-Durchläufen nicht mehr weiter berücksichtigt. In diesem Fall kann man von einem tatsächlichen „Vergessen" des IslEnquirers sprechen. Da die Daten jedoch nicht gelöscht werden, kann das System sich bei Bedarf aber genauso wieder an eine Person „erinnern".

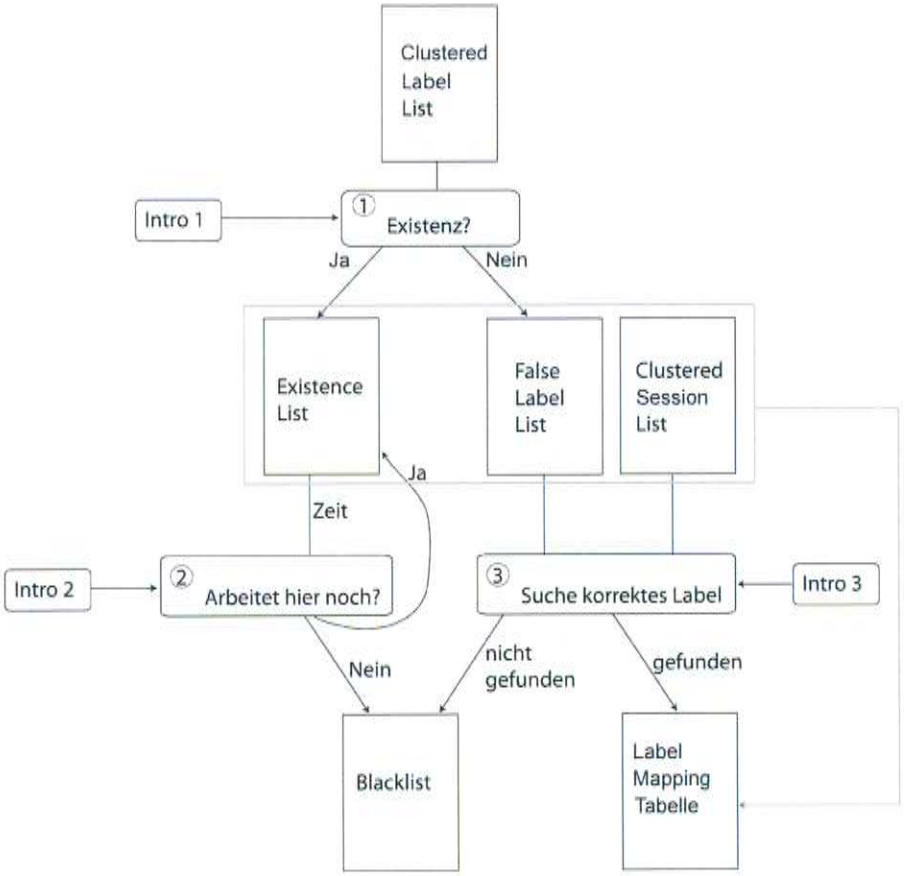

Abbildung 4.5: Schematische Darstellung des Aufbaus der Korrektur-Dialoge

Da der Existenz-Dialog fehlerhaft gelabelte Sessions nur implizit auflösen kann, ist noch ein dritter Korrektur-Dialog denkbar, der eine solche Label-Auflösung explizit macht. In diesem Fall würden Label, die innerhalb eines Clusters einen Labelkonflikt erzeugen, im Dialog direkt gegenüber gestellt. Dieses Vorgehen ist jedoch aus mehreren Gründen problematisch. Zum einen muss anhand der Label nicht zwingend klar sein, um wen es sich in den zugrundeliegenden Aufnahmen handelt, und zum anderen kön-

nen sämtliche vorgeschlagenen Label falsch sein, was dann wiederum eine explizite Angabe eines alternativen Labels erfordern würde. Eine solche explizite Angabe birgt auf der Spracherkennungsseite jedoch erneut die gleichen Probleme, die ursächlich für die Fehler in der Wissensbasis verantwortlich sind.

Da dieser dritte Korrektur-Dialog derart problematisch ist, haben wir ihn für unsere weiteren Evaluationen ausgeklammert und stattdessen untersucht, inwiefern die gemeinsamen Informationen aus Clustering und Existenz-Dialog genügen, ein zuverlässiges Label-Mapping zu garantieren. In Abbildung 4.5 entspricht dieses Vorgehen dem grauen Pfad, der ein Überspringen des dritten Korrektur-Dialogs ermöglicht.

4.3.2.1 Sortierung der Label nach Priorität

Möchte man die zu erfragenden Label bzw. Namen nach Priorität sortieren, ist es naheliegend Informationen zu nutzen, die uns das Clustering zur Verfügung stellt. Das Problem dabei ist, dass das Clustering die Cluster von Sessions und nicht etwa von Labeln ermittelt. Natürlich besteht zwischen diesen beiden ein kausaler Zusammenhang. Dennoch bedarf es einiger Schritte, um die Session-Cluster-Information für die Label-Sortierung nutzbar zu machen.

Zunächst gehen wir von der Clustered-Session-List (CSL) aus, die, wie in Kapitel 4.3.1 beschrieben, für jedes Cluster die Label-Verteilungen als Hypothesen-Liste aufführt. Mit dieser haben wir folglich eine Zuordnung zwischen Cluster, Session und möglichen Labeln. Ausgehend von einem beliebigen Label wählen wir zunächst die Sessions, die dieses Label tragen, und suchen die dazugehörigen Cluster, um ihre Clustered-Session-Lists zu einer Clustered-Label-List (CLL) zu vereinen. Zur Anschaulichkeit ein Beispiel aus Abbildung 4.4. Gehen wir von dem Label $\diamond$ aus, dann tragen drei Sessions dieses Label. Zwei in Cluster C und eins in Cluster D. Vereint lautet die Verteilung innerhalb der beiden Cluster: 60% $\diamond$, 15% $\oslash$, 15% $\times$, was damit der Clustered-Label-List für das Label $\diamond$ entspricht.

Auf diese Weise wird nun für jedes Label innerhalb der Wissensbasis eine Clustered-Label-List generiert. Sie ist eine Art Zusammenfassung der verschiedenen Clustered-Session-Lists und drückt so letztlich den Grad der Label-Verwechslung bei Sessions dieses Labels aus.

Für die Sortierung werden schließlich folgende vier Sortierkriterien verwendet:

- **Häufigkeit**: Anzahl der Sessions mit entsprechendem Label.
- **Unsicherheit**: Entropie auf der Clustered-Session-List des Clusters, das die meisten Sessions mit diesem Label enthält.
- **Verwechslungsgrad**: Anzahl Label in Clustered-Label-List.
- **Relevanz**: Zeitspanne seit letzter Session mit diesem Label.

Die Liste der Label wird für jedes dieser Kriterien separat sortiert und durch eine Standard-Ranglisten-Fusion vereinigt, wobei ein niedriger Rang eine hohe Priorität ausdrückt. Mathematisch lässt sich die Rangfolgen der Sortierungskriterien demnach, wie folgt ausdrücken:

$$rank_{\text{Häufigkeit}}(A) = \sum_{j=0}^{J} freq_j(A) \tag{4.4}$$

$$rank_{\text{Unsicherheit}}(A) = entropy(argmax_j freq_j(A))^{-1} \tag{4.5}$$

$$= -(\sum_{L \in labels_j} gewicht_j(L) \cdot log_2(gewicht_j(L)))^{-1} \tag{4.6}$$

$$rank_{\text{Verwechslungsgrad}}(A) = |CLL_{label}(A)|_l^{-1} \tag{4.7}$$

$$rank_{\text{Relevanz}}(A) = lastseen(A)^{-1} \tag{4.8}$$

Wobei die verwendeten Funktionen, wie folgt zu verstehen sind:

$$\begin{aligned}
cluster_j &= \text{Cluster mit Index } j \\
|cluster_j|_s &= \text{Anzahl Session in } cluster_j \\
freq_j(A) &= \text{Anzahl Session mit Label } A \text{ in } cluster_j \\
gewicht_j(A) &= \frac{freq_j(A)}{|cluster_j|_s} \\
labels_j &= \{L \in labels | freq_j(L) > 0\} \\
CLL_{label}(A) &= \{L \in labels | \exists j : freq_j(A) \cdot freq_j(L) > 0\}
\end{aligned}$$

4.3.2.2 Existenz-Dialog

Ziel des Existenz-Dialogs ist es bei einer trusted-Person zu erfragen, ob ein gegebenes Label einem Namen einer bekannten Person entspricht. Angemerkt sei, dass die Existenz eines Labels als Namen einer Person nicht zwingend bedeutet, dass das entsprechende Label auch korrekt zugeordnet ist. Denn das Label muss sich nicht zwingend auf die richtige Person beziehen. Beispielsweise bei einem Fehler des dritten Typus, wäre dies nicht der Fall. Je nach eingestellten Parameter erfragt der IslEnquirer eine oder mehrere fragliche Label innerhalb eines Existenz-Dialogs. Um Fehler bei der Spracherkennung zu vermeiden, werden die Antworten jeweils durch einen Bestätigungsschritt abgesichert. Die Existenz-Information wird dann in einem späteren Schritt (siehe Kapitel 4.3.3) für die Auflösung der Labelkonflikte genutzt, in dem entsprechende Label-Mappings abgeleitet werden. Ein Beispiel für einen Existenz-Dialog findet sich in Tabelle 4.1.

System:	Do you know someone called „HartwigHartwig“?
User:	Sorry, i don't know anyone called „Hartwig Hartwig“.
System:	Did you say you do not know „HartwigHartwig“?
User:	Yes, that's right!
System:	Thank you! Now one more question.
	Do you know a person called „PhilippGroße“?
User:	Indeed, I do.
System:	I understood you sad you know „PhilippGroße“. Is that correct?
User:	Yes!
System:	Thanks for the information. I will put it into my database.

Tabelle 4.1: Beispiel für einen Existenz-Dialog, der zwei Label überprüft

4.3.3 Dialogbasiertes Mapping

Ausgehend von der Verteilung der Sessions innerhalb der Cluster verbunden mit der Information über die Existenz verwendeter Session-Label als Namen, lassen sich etwas ausgefeiltere Label-Mapping-Regeln ableiten, als dies in Kapitel 4.3.1 nur dem der Clustering-Information möglich war.

Zu Veranschaulichung zeigt Abbildung 4.6, wie das Wissen um die Existenz der Label als Namen, helfen kann die fehlerhaften Label aus Abbildung 4.4 aufzulösen. Die Farbe grün steht für eine Bestätigung, während rot für eine Zurückweisung des Labels innerhalb des Existenz-Dialogs

steht. Für **schwarz** gefärbte Label in der Abbildung wurden im Dialog bis zu dem Zeitpunkt noch keine Informationen gesammelt.

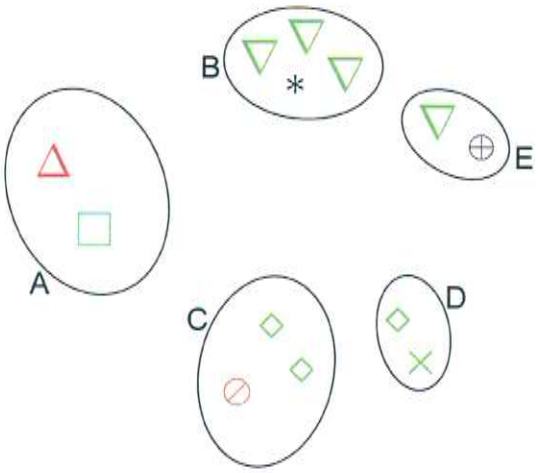

Abbildung 4.6: Schematische Darstellung der finalen Cluster und der in ihnen vertretenen Session-Label nach Existenzfragen im Korrektur-Dialog

Anhand des Clusters A zeigt sich recht deutlich der Mehrwert, der durch den Existenz-Dialog entsteht. War die korrekte Auflösung des Labelkonflikts innerhalb dieses Clusters mit dem clusterbasierten Mapping noch reiner Zufall, so scheint dies mit der Existenz-Information nun deutlich leichter.

Und auch bei den Clustern D und E hilft die Zusatzinformation durch den Dialog. Im Fall des Clusters D verstärkt sich der beobachtete Labelkonflikt zu einem unauflösbaren Labelkonflikt. Grund hierfür ist, dass mehr als nur ein Label innerhalb des Clusters als existierend bestätigt wurden, was wiederum bedeutet, dass entweder ein Fusionsfehler vorliegt, oder ein Fehler des dritten Typus. Auch wenn nicht klar ist worin die Ursache für den verstärkten Labelkonflikt besteht, kann doch abgeleitet werden, dass die Session-Label innerhalb des Clusters fraglich sind. Dies hilft die Fehler auf der Wissensbasis zu reduzieren.

Im Fall des Clusters E hilft uns die Dialog-Information ebenso, um Fehler auf der Wissensbasis zu reduzieren. Aber es wäre voreilig zu glauben, man könne aufgrund der Existenz-Bestätigung des Labels ▽ das Label ⊕ darauf mappen. Ein Blick auf Abbild 4.3 offenbart, dass ein solches Label-Mapping ein Fehler wäre. Um solche Fehler zu vermeiden, betrachten wir

neben der Existenz-Dialog-Information auch die Verteilung der Label auf die einzelnen Cluster.

Grundsätzlich basiert das von uns verwendete dialogbasierte Mapping-Verfahren auf Fallunterscheidungen, in Abhängigkeit der Label-Bestätigungen innerhalb des Clusters. Die im Folgenden verwendete Farbnotation wird analog zu der in Abbildung 4.6 verwendet.

1. **Mehr als ein grün markiertes Label im Cluster der Session.** In diesem Fall kann das Session-Label nicht aufgelöst werden, weswegen Sessions innerhalb solch fraglicher Cluster aus Sicherheitsgründen mit 'Remove' markiert werden, statt sie zu mappen. Beispielsweise für die Sessions in Cluster D in Abbildung 4.6 wäre dies der anzuwendende Fall.

2. **Genau ein grün markiertes Label im Cluster der Session.** In diesem Fall wird geprüft, ob das grün markierte Label in keinem Cluster häufiger vertreten ist, als in dem Cluster der vorliegenden Session. Ist dies der Fall, wird das vorliegende Session-Label auf das grün markierte Label gemappt. Dies entspricht dem Vorgehen für die Sessions in Cluster A, B und C. Andernfalls wird aus Sicherheitsgründen, um Fehlmappings zu vermeiden, angenommen, dass es sich bei dem grün markierten Label in dem Cluster der Session, um einen Fehler des dritten Typus handelt, oder dass das Cluster durch eine Fehlfusion entstanden ist. Dies mag nicht zwingend der Fall sein, da jedoch die Daten eines Cluster pro Person i.d.R. genügen, können weitere Sessions der Person bedenkenlos mit 'Remove' markiert werden, wenn dies im Zweifelsfall der Fehlerreduktion auf der Wissensbasis dient. Alle weiteren Sessions des Clusters werden ebenso mit 'Remove' markiert, da das einzig bestätigte Label fraglich erscheint. Dies entspricht dem Vorgehen für die Sessions in Cluster E.

3. **Alle Label im Cluster der Session sind rot markiert.** In diesem Fall kann das Session-Label nicht aufgelöst werden, weshalb die Session mit 'Remove' markiert wird, statt sie zu mappen. Auf diese Weise wird die Wissensbasis zumindest um ein bekanntlich fehlerhaftes Label bereinigt.

4. **Kein Label im Cluster der Session ist grün markiert.**
   a) **Fragliches Session-Label ist rot markiert.** Da das Session-Label in diesem Fall nicht aufgelöst werden kann, aber bereits

bekannt ist, dass es sich bei dem bisher verwendeten Label um einen nicht existierenden Namen handelt, wird die Session vorläufig mit 'Remove' markiert, statt sie zu mappen. So wird die Wissensbasis, zumindest vorläufig, um ein bekanntlich fehlerhaftes Label bereinigt. Wenn im späteren Verlauf eines der anderen Label im Cluster als existierend bestätigt wird, lässt sich das Session-Label im Zweifelsfall sogar richtig auflösen.

b) **Fragliches Session-Label ist nicht rot markiert.** In diesem Fall greifen wir auf das clusterbasiete Mapping-Verfahren als Backup-Methode zurück und haben damit zumindest eine gewisse Chance die Session trotz fehlender Dialog-Informationen korrekt aufzulösen.

Am Ende der Fallunterscheidung wurde das Label der zu untersuchenden Session entweder auf ein (neues) Label gemappt, oder aber mit 'Remove' markiert. Im letzteren Fall bedeutet dies, dass die fragliche Session nicht weiter für das Training der FaceID/VoiceID verwendet werden sollte und dass das ursprüngliche Label im Zweifelsfall aus der Wissensbasis genommen wird. Es bedeutet allerdings nicht zwingend, dass die Session auch ganz aus dem Datenkorpus genommen wird, denn es ist nicht ausgeschlossen, dass die Session in späteren Cluster-Durchläufen dank neu aufgezeichneter Sessions doch noch richtig aufgelöst werden kann. Hier ist es eine Frage der Systemeinstellungen wie lange man solche Sessions bereit hält und ab welchem Zeitpunkt man auf sie für weitere Clustering-Durchläufe verzichtet.

5 Experimente und Ergebnisse

In folgendem Kapitel werden die verschiedenen Evaluationen und Ergebnisse, die im Rahmen der hier vorliegenden Arbeit durchgeführt wurden, vorgestellt. Zu diesem Zweck unterteilt sich das Kapitel in mehrere Teile. Der erste Teil stellt die verschiedenen Evaluationsmetriken vor, die in der darauf folgenden Teilen verwendet werden, um die Qualität der Wissensbasis bzw. die Verbesserung dieser zu bewerten. Der zweite Teil stellt die Daten bzw. Datensets vor, die für das Training bzw. die Evaluation, des in dieser Arbeit vorgestellten Systems, verwendet wurden. Teil drei beschreibt das Training und die Auswahl der Konfidenzmerkmale, sowie die Evaluation dieser. Dabei wird die Konfidenz nicht nur in ihrer Eigenschaft als Abbruchbedingung sondern auch ihre Eignung als adaptives Distanzmaß des Clustering-Verfahrens evaluiert. Teil vier befasst sich schließlich mit der Evaluation der Mapping-Verfahren und untersucht insbesondere die Wissensbasis-Bereinigung dieser.

5.1 Evaluationsmetriken

In dem nun folgenden Unterkapiteln werden eine Reihe verschiedener Evaluationsmetriken zur Evaluation der Wissensbasis Qualität genutzt, die nun gebündelt vorgestellt werden.

Zwei der im Bereich *Information Retrieval* und Klassifikator Bewertung gängigen Evaluationsmaße sind Recall und Precision. In unserem Fall verwenden wir die Beiden, um die Qualität, der in der Wissensbasis geführten Label, zu erfassen. Die Precision wird dabei vor allem durch Fehler des ersten, zweiten und vierten Typus[1] beeinflusst, während der dritte Fehlertyp in dieser Betrachtung nicht erfasst werden kann. Der Recall hingegen drückt die Trefferquote der Label aus bzw. wie gut die Groundtruth durch die Label abgebildet wird und bringt damit die nicht auflösbaren Label (siehe 5.1) in die Evaluation mit ein. Mathematisch ausgedrückt sind die

[1] für eine Beschreibung der Fehlertypen siehe Kapitel 2.4.1

beiden Werte wie folgt definiert:

$$\text{Precision} = \frac{\text{Anzahl korrekter Label in der Wissensbasis}}{\text{Anzahl Label in der Wissensbasis}} \tag{5.1}$$

$$\text{Recall} = \frac{\text{Anzahl korrekter Label in der Wissensbasis}}{\text{Anzahl Personen in Groundtruth}} \tag{5.2}$$

$$F1 = \frac{1}{0.5 \cdot (\frac{1}{Precision} + \frac{1}{Recall})} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.3}$$

Wobei „korrektes“ Label in den Gleichungen nicht zwingend bedeutet, dass das Label die richtige Person beschreibt, sondern lediglich, dass das Label mindestens einmal als Name in der Groundtruth aufgeführt wird. Dieser Sachverhalt ist auch die Ursache, wieso Recall und Precision keinen Fehler des dritten Typus anzeigen können.

Um den Recall und die Precision in einem Wert zusammenzufassen wird häufig auch das so genannte „F-Measure“ (F1) verwendet, welches das harmonische Mittel der beiden darstellt. Für unsere Evaluation verwenden wir teilweise auch die inverse Darstellung des F-Measures und der Percision, bezeichnet als F-Measure Error Rate (F1ER) und Entry Error Rate (EER). Definiert sind diese inversen Maße durch folgende Gleichungen:

$$\text{F1ER} = 1 - \text{F1} \tag{5.4}$$

$$\text{EER} = 1 - \text{Precision} \tag{5.5}$$

Da der Recall und die Precision, wie bereits erwähnt, Fehler des dritten Typus nicht erfassen können, verwenden wir noch ein weiteres Evaluationsmaß. Die Session Error Rate (SER) bzw. ihre Inverse, die Session Label Korrekt Rate. Diese drücken das Verhältnis zwischen korrekt gelabelten Sessions und fehlerhaft gelabelten Sessions in der Wissensbasis aus. Mathematisch ausgedrückt ist die SER demnach definiert durch:

$$SER = \frac{\text{Anzahl falsch gelabelter Sessions in der Wissensbasis}}{\text{Anzahl Sessions in der Wissensbasis}} \tag{5.6}$$

5.2 Beschreibung der Daten

Zum Zweck der Evaluation wurden drei Daten-Sets erstellt, die jeweils einen Ausschnitt der Wissensbasis des IslEnquirers darstellen. Das Trainingsset wurde, wie der Name schon andeutet, für das Training der Logit-Koeffizienten der Logistischen Regression geschaffen. Wohingegen das Set1 das Hauptevaluationsset darstellt. Es umfasst den Großteil der in der Wissensbasis vorliegenden Sessions, wobei bewusst auf Sessions von Personen verzichtet wurde, die nur ein einziges Mal mit dem System interagiert haben. Ein solches Vorgehen ist insofern zulässig, da solche Personen fast ausschließlich Gäste des Instituts sind, die nicht in der Wissensbasis abgebildet werden müssen. Außerdem enthält das Set1, zum Zweck der sauberen Trennung von Training und Evaluation, keine der im Training verwendeten Sessions. Beim zweiten Evaluationsset (Set2) handelt es sich um eine Untermenge des ersten Evaluationssets (Set1), welche um sämtliche nicht auflösbaren Sessions bereinigt wurde. Sinn des zweiten Evaluationssets ist es, eventuelle Verzerrungen, die durch die Existenz nicht auflösbarer Sessions bei der Evaluation mit Set1 entstehen mögen, zu vermeiden.

Trainings-Set				*Fehlertyp*		
	Total	*fehlerhafte Label*	*nicht auflösbar*	*1.*	*2.*	*3.*
Session	59	18 (30.5%)	-	2	14	2
Label	26	15 (57.7%)	-	1	14	1
Personen	17	9 (52.9%)	-	1	9	1
Set1				*Fehlertyp*		
	Total	*fehlerhafte Label*	*nicht auflösbar*	*1.*	*2.*	*3.*
Session	106	31 (29.3%)	15 (48.4%)	8	21	2
Label	45	23 (51.1%)	8 (34.8%)	4	17	2
Personen	27	18 (66.7%)	5 (27.8%)	4	14	2
Set2				*Fehlertyp*		
	Total	*fehlerhafte Label*	*nicht auflösbar*	*1.*	*2.*	*3.*
Session	91	16 (17.6%)	0 (0%)	3	12	1
Label	37	15 (51.1%)	0 (0%)	2	12	1
Personen	22	12 (54.7%)	0 (0%)	2	10	1

Tabelle 5.1: Beschreibung der Sets

Die Tabelle 5.1 zeigt die Parameter der Sets. Dabei wird für jedes Set die Anzahl der in ihr enthaltenen Sessions, Personen und Label in der Spalte

„*Total*" aufgeführt. In der zweite Spalte „*fehlerhafte Label*" wird die entsprechende Anzahl bzw. der prozentuale Anteil an fehlerhaften Einträgen angegeben. Während in der dritten Spalte „*nicht auflösbar*" die Anzahl bzw. der prozentuale Anteil der fehlerhaften Einträge aufgeführt wird, die nicht durch ein Label-Mapping auflösbar sind. Dies ist immer dann der Fall, wenn für die betroffene Person kein korrektes Label in der Wissensbasis vorliegt. Damit ist die Bereinigung der Wissensbasis in diesen Fällen nicht durch ein Label-Mapping, sondern nur durch eine 'Remove'-Markierung der entsprechenden Session möglich, wie sie in Kapitel 4.3.3 eingeführt wurde.

Die letzten drei Spalten führen die verschieden Fehlertypen, entsprechend der Definition aus Kapitel 2.4.1, auf. Wobei auf eine Aufführung der Fehler des vierten Typus verzichtet wurde, da diese, wie in Kapitel 4.1 dargestellt, nicht durch Daten-Analyse behoben werden können und daher bei den folgenden Evaluationen vernachlässigt wurden.

Angemerkt sei, dass sich die Anzahl der aufgeführten Fehlertypen nicht immer sauber aufsummieren lassen. Beispielsweise werden bei dem Trainingsset 9 Personen mit fehlerhaften Labeln aufgeführt, wobei die Summe der Fehlertypen eins, zwei und drei sich auf 11 summiert. Die Ursache ist, dass eine Person in der Wissensbasis mit mehreren Labeln vertreten sein kann und auf diese Weise Fehler in mehreren Rubriken pro Person möglich sind. Ähnliches gilt für die Zeile der Label. Hier produzieren bei dem Trainingsset 15 Label 16 verschiedene Fehler. Der Grund hierfür findet sich im Fehler des dritten Typus, bei dem ein korrektes Label einen Fehler produziert, obwohl es nicht bei den fehlerhaften Labeln mit aufgeführt wird.

Abbildung 5.1 zeigt die aus der Einleitung 1.1 vertraute Darstellung der Wissensbasis Verunreinigung für Set1 im zeitlichen Verlauf. Sie veranschaulicht, wie schon Abbildung 1.1, die immer größer werdende Divergenz zwischen Anzahl gelernter Label und bekannter Personen. Neben der inversen Session Error Rate (siehe Gleichung 5.6) führt sie auch den Recall und die Precision, sowie das F-Measure für die Label des Sets auf (siehe Gleichungen 5.1-5.3). Hier sieht man deutlich, wie diese Kurven mit Anstieg der Divergenz zwischen Label und Personen abfallen. Insbesondere der Precision-Wert der Label ist auffällig, fällt er doch gegen Ende der Grafik auf knapp unter 50%; was, wie wir auch schon Tabelle 5.1 entnehmen konnten, bedeutet, dass gegen Ende der beobachteten Zeitspanne über 50% aller Label fehlerhaft in dem Set vorliegen. Dieser Wert zeigt sehr deutlich welchen Einfluss die fehlgelernten Label auf die Qualität der Wissensbasis haben.

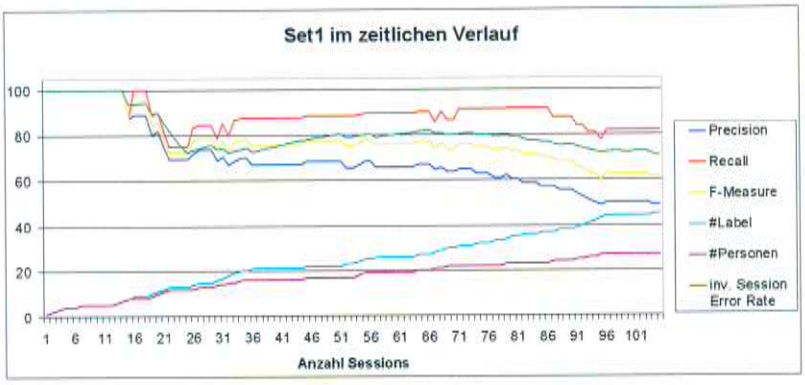

Abbildung 5.1: „Verunreinigung" des Set1 im zeitlichen Verlauf ohne Datenbereinigung

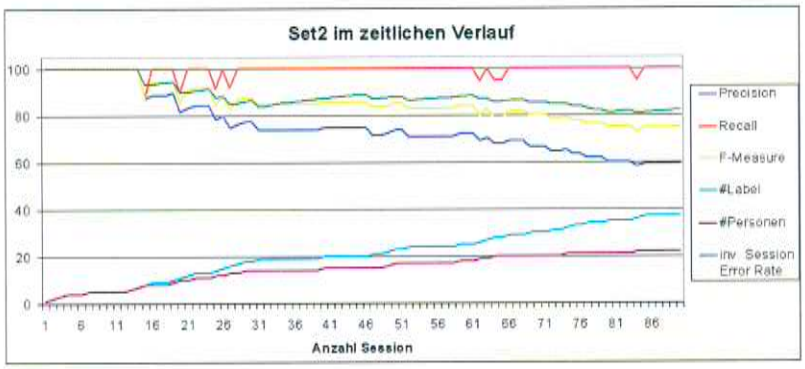

Abbildung 5.2: „Verunreinigung" des Set2 im zeitlichen Verlauf ohne Datenbereinigung

Abbildung 5.2 zeigt selbige Divergenz und Kurvenentwicklung für das zweite Evaluationsset (Set2). Anzumerken sind, neben den bereits für Abbildung 5.1 geschilderten Tendenzen, die kurzen Einbrüche beim Recall-Wert. Ein solcher Einbruch ist immer dann zu verzeichnen, wenn der IslEnquirer eine neue Person kennenlernt, deren Namen jedoch durch ein falsches Label abspeichert. Dies wird dann, wie wir anhand des wieder nach oben schnellend Recall-Wert erkennen können, in einem kurz darauf folgenden Dialog wieder korrigiert, wenn der IslEnquirer den Namen doch noch richtig lernt. Anders als Set1 enthält Set2 folglich keine Personen, deren Namen das System nicht doch noch richtig lernt. Dieser Umstand zeigt sich recht deutlich anhand des Recall-Werts.

5.3 Distanzmaße & Konfidenz

5.3.1 Training der Konfidenz

Für das Training eines robusten Klassifikators, bzw. im Fall der logistischen Regression guter Logit-Koeffizienten, ist die Vielfalt der verwendeten Trainingsdaten zentral. Da unser Trainingsset jedoch nur 59 Sessions enthält und somit auch nur maximal 58 Cluster Fusionen, in dem von uns beschrieben Clusterverfahren, zulässt, bedarf es einiger Anpassungen, um dennoch ausreichend Trainingsdaten für die Logit-Koeffizienten zu erhalten. Die bei 58 Clusterfusionen gesammelten Konfidenzmerkmale würden prinzipiell für ein Training ausreichen. Das so trainierte Modell (die ermittelten Koeffizienten) würde dann jedoch nicht über die Trainingsdaten hinweg generalisieren. Sprich die ermittelten Koeffizienten der Logistischen Regression würden die Trainingsset spezifischen Parameter, wie etwa Anzahl Session, Anzahl Labelfehler, etc. implizit mit modellieren.

Um dies zu vermeiden, haben wir uns bemüht eine möglichst große Varianz innerhalb des Trainingskorpus zu generieren. Zu diesem Zweck haben wir einen doppel-randomisierten Auswahl-Prozess der Training-Sessions verwendet. Genau genommen haben wir den Clustering-Prozess des Trainingsset nicht nur einmal von einzelnen Session-Clustern zu einem alle Daten umfassenden Cluster fusionieren lassen, sondern diesen Prozess über hundert Mal wiederholt, wobei in jedem dieser Durchläufe der verwendete Trainingskorpus leicht abgewandelt wurde.

Doppelt-randomisiert ist diese Trainingskorpus-Abwandlung insofern, als dass sowohl die Anzahl verwendeter Sessions, als auch die Auswahl der Sessions dem Zufall überlassen wurde. Wobei die Anzahl zu verwendender Sessions innerhalb der Grenzen von 45% bis 100% des Trainingsset festgelegt wurde, um sicher zu stellen, dass die verwendeten Korpura nicht zu klein werden und damit zu wenige Clusterfusionen ermöglichen.

Wenn man es genau nimmt ist der Trainings-Prozess sogar Tripel-randomisiert, da die DCT-Vektoren bei Sessions mit mehr als 400 Bildvektoren, wie in Kapitel 4.2.2 beschrieben, per Zufall ausgewählt werden.

Da vor dem Training der Logit-Koeffizienten keine valide Konfidenz berechnet werden kann, werden, während des Trainings-Prozesses, die Fusionen der Cluster basierend auf den „normalen" Distanzmaßen vollzogen, statt das adaptive Distanzmaß der Konfidenz zu verwenden. Nach dem die Logit-Koeffizienten jedoch durch das Training der logistischen Regression bestimmt wurden, wird die Konfidenz, wie in Kapitel 4.2.3 dargestellt, als adaptives Distanzmaß genutzt.

5.3.2 Auswahl der Konfidenz-Merkmale

Während des, in Kapitel 5.3.1 dargestellten, Trainings-Prozesses werden für alle durchgeführten Cluster-Fusionen neben einem Wahrheitswerts über die Korrektheit der Fusion, folgende Merkmale in eine Datei geschrieben:

1. Die absolute Distanz zwischen den fusionierten Clustern
2. Die relative Distanz, gemäß Gleichung 4.2 aus 1. abgeleitet
3. Die Anzahl an Sessions in Cluster 1
4. Die Anzahl an Sessions in Cluster 2
5. Die durchschnittliche Editier-Distanz zwischen den fusionierten Clustern

Jedes dieser hier aufgeführten Merkmale eignet sich, wie in 4.2.3 erläutert, als potenzielles Konfidenzmerkmal. Ziel ist es nun das Merkmal bzw. die Merkmalskombination auszuwählen, die sich am Besten für die Konfidenzbestimmung eignet. Zu diesem Zweck verwenden wir den ROC-Graph [Faw03], der die Sensitivität[2] der Falsch-Positiv-Rate[3] des Klassifikators - in unserem Fall der mittels logistischer Regression ermittelten Konfidenz - gegenüberstellt.

Abbildung 5.3 zeigt einen Ausschnitt des ROC-Graphs, der bei der Wahl der Konfidenzmerkmale bei der centroidbasierten Euklid-Distanz zum Einsatz kam. Je weiter sich der Wert in der Graphik der linken oberen Ecke annähert, desto besser, denn dies bedeutet das Konfidenzmerkmal (bzw. die Konfidenzmerkmalskombination) unterscheidet korrekte von fehlerhaften Fusionen richtig.

Der Abbildung 5.3 kann man entnehmen, dass es für die Konfidenzbestimmung keinen qualitativen Unterschied macht, ob die Distanz absolut oder relativ angegeben wird. Weiter zeigt sich, dass die intra-Cluster-Distanz in Form der Cluster-Größen (ClusterSize) eine höhere Erkennungsrate erreicht, als die beiden centroidbasierten Distanzmaße (absDist, relDist), jedoch auch eine höhere Fehlerquote aufweist. Die Kombination beider Distanzansätze, also der inter-Cluster Distanz in Form der centroidbasierten Distanz und der intra-Cluster-Distanz in Form der Cluster-Größen, ergänzen sich hervorragend. Die Kombination weist sowohl bei

[2] True-Positiv-Rate (TP-Rate)
[3] False-Positiv-Rate (FP-Rate)

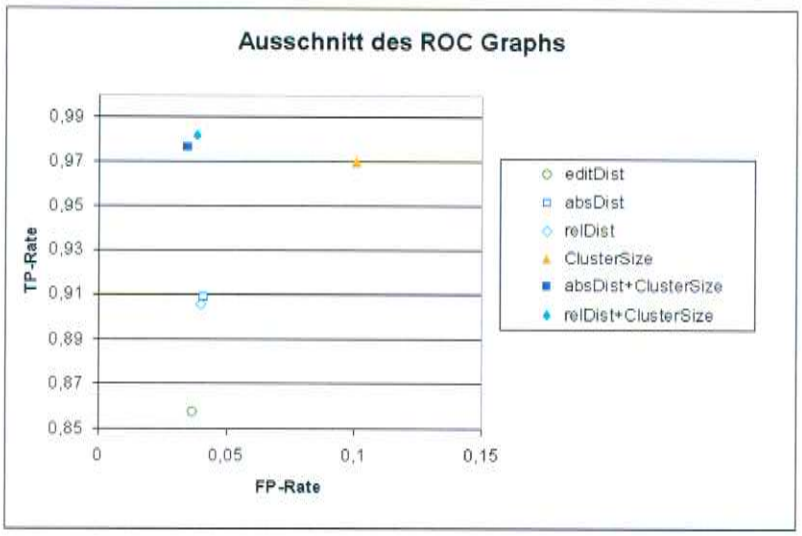

Abbildung 5.3: Konfidenz-Merkmal-Vergleich im ROC Graph

der TP-Rate, als auch bei der FP-Rate bessere Werte auf, als die in ihr vereinten Merkmale für sich.

Zusätzlich dazu zeigt der ROC-Graph auch die Position der durchschnittlichen Editier-Distanz als Konfidenzmaß. Von einer Kombination dieser mit einer der anderen Merkmale haben wir in dieser Abbildung abgesehen, denn dies entspräche in etwa dem Vorgehen der Hybriden-Distanz, welche wir in Kapitel 5.3.3 genauer analysieren werden.

Da sich die beiden Merkmalskombinationen „absDist + ClusterSize" und „relDist + ClusterSize" bei der Konfidenzbestimmung qualitativ nicht wirklich unterscheiden, haben wir uns schließlich für ersteres entschieden, da diese etwas weniger rechenaufwendig ist. Letztlich ist die Wahl zwischen diesen beiden Alternativen jedoch relativ willkürlich.

Logit-Koeffizienten				
Konfidenzmerkmal	i	Koeffizient β_i	$\varnothing$ Merkmalswert d_i	$\beta_i \cdot d_i$
	0	-34.1692		
absDist	1	6.2088	3.9	24.3
Size-of-Cluster1	2	1.5807	9.7	15.3
Size-of-Cluster2	3	2.1548	1.9	4.0

Tabelle 5.2: Logit-Koeffizienten für Konfidenzen bei centroidbasierter Euklid-Distanz

Tabelle 5.2 zeigt die für diese Merkmalskombination bestimmten Logit-Koeffizienten, wobei jeweils auch der Durchschnittswert des jeweiligen Merkmals angegeben wird. Der letzten Spalte, welche die Multiplikation des Koeffizienten mit dem Durchschnittswert zeigt, kann man etwa die Gewichtung der Merkmale entnehmen. Dabei fällt auf, dass die Größe des erstens Clusters deutlich stärker berücksichtigt wird, als die des zweiten Clusters. Dies lässt sich insofern erklären, als dass das erste Cluster durch die Sortierung der Cluster-Indizes immer das Größere von beiden Clustern ist.

Logit-Koeffizienten				
Konfidenzmerkmal	i	Koeffizient β_i	⌀ Merkmalswert d_i	$\beta_i \cdot d_i$
	0	-21.7382		
absDist	1	3.7519	4.0	15.1
Size-of-Cluster1	2	1.7949	10.0	18.0
Size-of-Cluster2	3	0.5788	1.7	1.0

Tabelle 5.3: Logit-Koeffizienten für Konfidenzen bei Hybrid-Distanz

Logit-Koeffizienten				
Konfidenzmerkmal	i	Koeffizient β_i	⌀ Merkmalswert d_i	$\beta_i \cdot d_i$
	0	-4.925		
absDist	1	0.2971	6.2	1.8
Size-of-Cluster1	2	0.9293	4.9	4.5
Size-of-Cluster2	3	0.5046	2.0	1.0

Tabelle 5.4: Logit-Koeffizienten für Konfidenzen bei Editier-Distanz

Logit-Koeffizienten				
Konfidenzmerkmal	i	Koeffizient β_i	⌀ Merkmalswert d_i	$\beta_i \cdot d_i$
	0	-55.5634		
absDist	1	5.2993	11.5	60.7
Size-of-Cluster1	2	1.5132	6.8	10.3
Size-of-Cluster2	3	0.9924	2.4	2.4

Tabelle 5.5: Logit-Koeffizienten für Konfidenzen bei durchschnittlicher inter-Cluster-Distanz

Analog zu dem soeben beschriebenen Vorgehen wurden auch für die anderen Distanzmaße passende Logit-Koeffizienten bestimmt, wobei die Tabellen 5.3, 5.4 und 5.5 die entsprechenden Werte ausweisen. Interessanter Weise hat sich bei den anderen Distanzmaßen ebenso die „absDist

+ ClusterSize" als Merkmalskombination durchgesetzt, was ein weiterer Hinweis dafür ist, wie gut sich die Merkmalskombination für die Konfidenzbestimmung eignet.

Studiert man die Tabellen genauer fällt auf, dass bei der Hybrid-Distanz sowie bei der durchschnittlichen inter-Cluster Distanz die Größe des zweiten Clusters kaum ins Gewicht fällt. Vermutlich hätte man bei beiden Distanzmaßen auf dieses dritte Merkmal verzichten können. Da die Evaluation für die Kombination der drei Merkmale jedoch nach wie vor etwas bessere Ergebnisse erzielte, haben wir uns dazu entschlossen die Kombination analog zu der centroidbasierten Euklid-Distanz mit allen drei Merkmalen beizubehalten.

5.3.3 Vergleich der Distanzmaße

Nachdem wir im vorangegangenen Kapitel die Logit-Koeffizienten ermittelt haben, evaluieren wir im Folgenden die Qualität der so bestimmten Konfidenz(-en) und vergleichen die Distanzmaße untereinander. Wie schon während des Trainings unterbrechen wir den Clustering-Prozess dabei nicht, sondern lassen den Prozess Cluster fusionieren bis schließlich nur noch ein alle Daten umfassendes Cluster entstanden ist. „Im Normalfall" würden wir den Prozess natürlich, wie in Kapitel 4.2.3 beschrieben, konfidenzgestützt unterbrechen. Für die Evaluation ermöglicht uns der Verzicht einer Unterbechung jedoch den gesamten Verlauf des Clustering-Prozesses zu beobachten.

Abbildung 5.4 zeigt den so durchlaufenen Clustering-Prozess für das erste Evaluationsset (Set1) bei der Verwendung der centroidbasierten Euklid-Distanz bzw. der auf ihrer Basis ermittelten Konfidenz. Die X-Achse weist dabei die Anzahl der Cluster aus, die zu Beginn des Clusterings der Anzahl an Sessions auf dem Datenset entspricht und mit jedem Clustering-Schritt abnimmt. Die Abbildung weist dabei folgende vier Kurven aus:

1. **Fusionsfehler-Rate**: Anzahl Sessions, deren in ihr enthaltenen Person von der des entsprechenden Clusters abweicht, geteilt durch die Anzahl Sessions im Datenkorpus. Das Cluster wird dabei analog zu der, in Kapitel 4.3.1 definierten, Clustered Session List (CSL) dem mehrheitlich in ihr vertretenen Person zugeordnet.
   Das Ansteigen dieser Kurve entspricht dem Zeitpunkt ab dem Cluster unterschiedlicher Personen fusioniert werden.

2. **Labelfehler-Detektions-Rate**: Anteil fehlerhafter Label, die mindestens mit einer ihr zugehörigen Session in einem Cluster einen Labelkonflikt erzeugen.
   Das Ansteigen dieser Kurve spiegelt den Grad der Labelfehler Detektion wieder, wobei sie spätestens zum Zeitpunkt, da alle Sessions in einem Cluster fusioniert wurden, einen Wert von 100% erreicht. Idealerweise erreicht das Clustering-Verfahren jedoch noch vor dem ersten Fusionsfehler eine 100% Labelfehler Detektion.

3. **Reverse-Konfidenz**: Statt die Konfidenz über die Richtigkeit der Fusion anzugeben, zeigt die Abbildung die Konfidenz in umgedrehter Darstellung. Reverse Konfidenz $= 100 -$ Konfidenz.
   Idealerweise fällt der Zeitpunkt des Anstiegs dieser Kurve mit dem der Fusionsfehler-Rate zusammen.

4. **Korrekt-fusionierte-Session-Rate**: Anzahl richtig fusionierter Sessions, geteilt durch die Anzahl Sessions im Datankorpus. Richtig fusioniert heißt in diesem Fall, das entsprechende Cluster enthält mehr als nur eine Session, jedoch nur Daten einer Person.
   Diese Kurve gibt den Grad der korrekt fusionierten Sessions an und damit auch den Grad der Abdeckung. Idealerweise würde diese Kurve einen Wert von 100% erreichen, bevor das Clustering-Verfahren, bedingt durch den Konfidenzwert, abgebrochen wird.

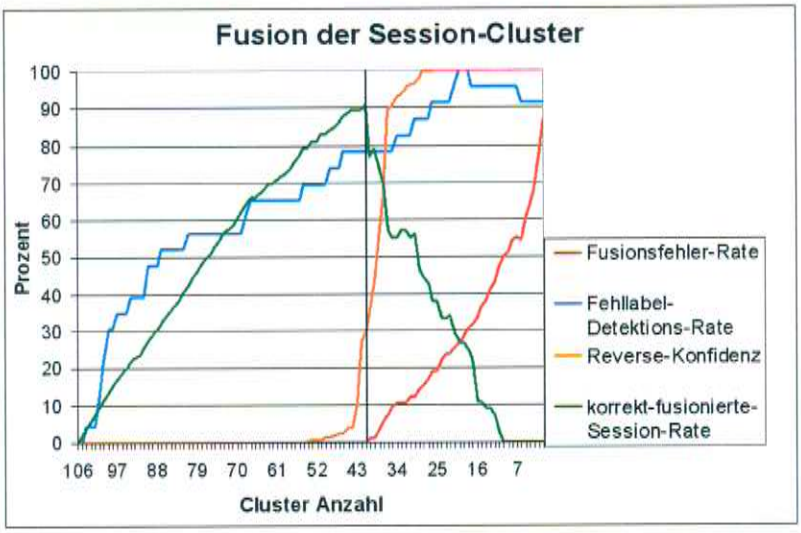

Abbildung 5.4: Cluster-Fusion bei Euklid-Distanz

Zusätzlich zu den Kurven, weist die Abbildung noch einen **senkrechten Balken** auf, der den idealen Zeitpunkt für einen Abbruch des Clustering-Verfahrens markieren soll.

Der Abbildung 5.4 lässt sich entnehmen, dass bis zum Zeitpunkt des idealen Abbruchs, also bevor die Fusionsfehler-Rate ansteigt, die Labelfehler-Detektions-Rate einen Wert von etwa 78.3% erreicht. Angesichts der Tatsache, dass wir der Tabelle 5.1 entnehmen können, dass nur etwa 65% der Label tatsächlich auflösbarer sind, bedeutet dies, dass wir in diesem Fall mehr fehlerhafte Label detektiere, als wir später tatsächlich durch ein clusterbasierte Mapping (siehe Kapitel 4.3.1) auflösen können. Das die Labelfehler-Detektions-Rate gegen Ende der Evaluation wieder auf unter 100% fällt, ist der Tatsache geschuldet, dass das Clustering-Verfahren auf Basis der 10% Outlier-Detektion bei übermässig großen Cluster anfängt die Daten ganzer Sessions aus dem Korpus zu entfernen. Da dies aber erst weit nach einem natürlichen Abbruchpunkt des iterativen Clustering-Verfahrens geschieht, stellt dies keinen grundsätzlichen Fehler des Verfahrens dar.

Da die Aufteilung der fehlerhaft gelabelten Sessions auf die korrekt fusionierten Cluster zufällig ist, bietet uns die korrekt-fusionierte-Session-Rate ein noch besseres Evaluationsmaß, denn diese gibt unabhängig von den Labeln an, wie viel Prozent der Sessions korrekt fusioniert wurden. Einem Spitzenwert von etwa 90% kann man daher entnehmen, dass zum Zeitpunkt des idealen Abbruchs nur 10% aller Sessions noch in Einzel-Session-Clustern verblieben sind, während alle anderen Sessions korrekt zugeordnet wurden. Umgekehrt bedeutet dies allerdings auch, dass für den Fall das fehlgelabelte Sessions unter den 10% nicht geclusterten Sessions sein sollten, diese eventuell nicht sicher aufgelöst werden können, da nicht geclusterte Sessions keine Labelkonflikte hervorrufen.

Wie bereits bei der Beschreibung der reversen Konfidenz angemerkt, steigt diese idealerweise mit Anstieg der Fusionsfehler-Rate, was wie wir der Abbildung 5.4 entnehmen können, für die Konfidenz des centroidbasierten Distanzmaßes der Fall ist. Ein Konfidenzschwellwert kann daher zuverlässig dazu genutzt werden, um den idealen Zeitpunkt für einen Abbruch des Clustering-Verfahrens zu bestimmen.

Analog zu Abbildung 5.4 wurden auch für die anderen Distanzmaße und deren Konfidenzen entsprechende Evaluationen auf Set1 durchgeführt, welche in den Abbildungen 5.5, 5.6 und 5.7 gezeigt werden.

Beim Vergleich zwischen den Distanzmaßen ist zu beachten, dass der in Kapitel 5.3.1 vorgestellte Trainings-Prozess, und somit auch der randomisierte Session-Auswahl-Prozess, für jedes zu untersuchende Distanzmaß

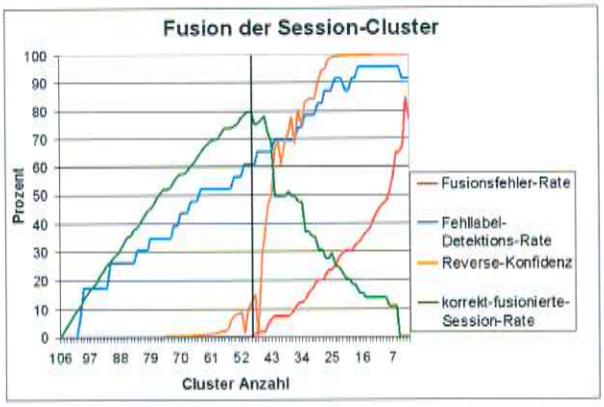

Abbildung 5.5: Cluster-Fusion bei Hybrid-Distanz

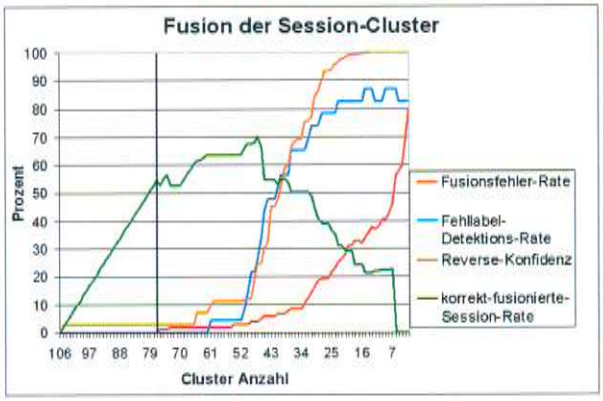

Abbildung 5.6: Cluster-Fusion bei durchschnittlicher Editier-Distanz

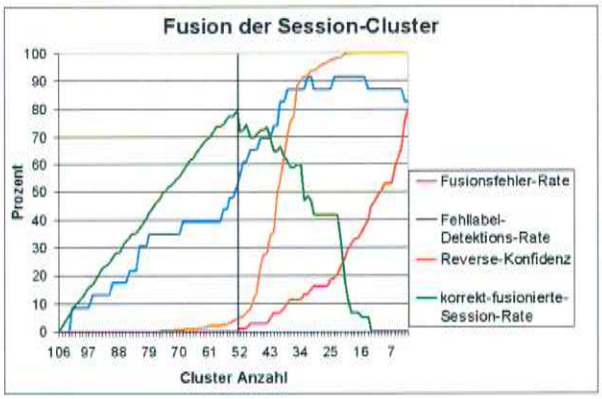

Abbildung 5.7: Cluster-Fusion bei durchschnittlicher inter-Cluster-Distanz

separat durchlaufen werden muss, da während des Trainings immer nur ein Distanzmaß als Grundlage für die Fusion dienen kann. Dieser Umstand hat zur Folge, dass die Trainingsdaten für jedes Distanzmaß variieren, was wiederum die Konfidenz für das jeweilige Distanzmaß beeinflussen kann und somit die Vergleichbarkeit der Distanzmaße untereinander erschwert. Wir haben jedoch Anlass zu der Annahme, dass die vielfache Wiederholung (hundert Durchläufe) diesen Effekt im statistischen Mittel ausgleicht. Dennoch sollte man sich im Klaren sein, dass durch die Randomisierung Schwankungen in der Qualität der ermittelten Konfidenzen möglich sind, was zur Folge hat, dass ein Vergleich zwischen den Distanzmaßen somit nur den Vergleich zwischen der Qualität der ermittelten Konfidenzen ermöglicht, nicht aber einen generellen Vergleich zwischen den Distanzmaßen.

Betrachet man die reverse Konfidenz des Hybriden-Distanzmaßes in Abbildung 5.5 so fällt auf, dass sie, ähnlich wie schon bei der centroidbasierten Euklid-Distanz, zum Zeitpunkt der ersten Fusionsfehler deutlich ansteigt und sich folglich ebenso für einen Schwellwert eignet, wie die Konfidenz in der ersten Evaluation. Auffällig ist jedoch auch, dass die Labelkonflik-Rate als auch die korrekt-fusionierte-Session-Rate keine so guten Werte liefern, wie in der Evaluation zuvor. Wieso die Hybride-Distanz trotz der zusätzlichen Editier-Distanz Information schlechtere Ergebnisse erzielt als der nur mit Euklid-Distanz arbeitende centroidbasierte Ansatz wird klar, wenn man sich die Ergebnisse der reinen Editier-Distanz in Abbildung 5.6 ansieht.

Dort zeigt sich, dass die Editier-Distanz deutliche Schwierigkeiten mit Fehlern des dritten Typus hat, denn obwohl die Editier-Distanz solcher Sessions minimal zu anderen Sessions dieses Labels ist, ist eine Fusion der Cluster dennoch falsch. Dies hat zur Folge, dass die Abbildung 5.6 als einziges den Fall aufweist in dem die Fusionsfehler-Rate vor der Fehllabel-Detektions-Rate steigt. Ein frühzeitiger Abbruch des Clustering-Prozesses, vor Anstieg der Fusionsfehler-Rate würde folglich nicht helfen, um fehlerhafte Session-Label aufzudecken.

Interessanterweise weist auch die wesentlich rechenaufwendigere durchschnittliche inter-Cluster-Distanz keine besseren Ergebnisse als die centroidbasierte Euklid-Distanz auf. Abbildung 5.8 zeigt die Distanzmaße und ihre korrekt-fusionierte-Session-Raten noch einmal im direkten Vergleich. Dabei fällt vor allem auf, wie weit die Spanne der idealen Abbruchzeitpunkte ist und dass nur die centroidbasierte Eukild-Distanz bzw. ihre Konfidenz eine 90% Abdeckung des Evaluationssets erreicht, was diese noch einmal deutlich als geeignetste der untersuchten Distanzmaße bestätigt.

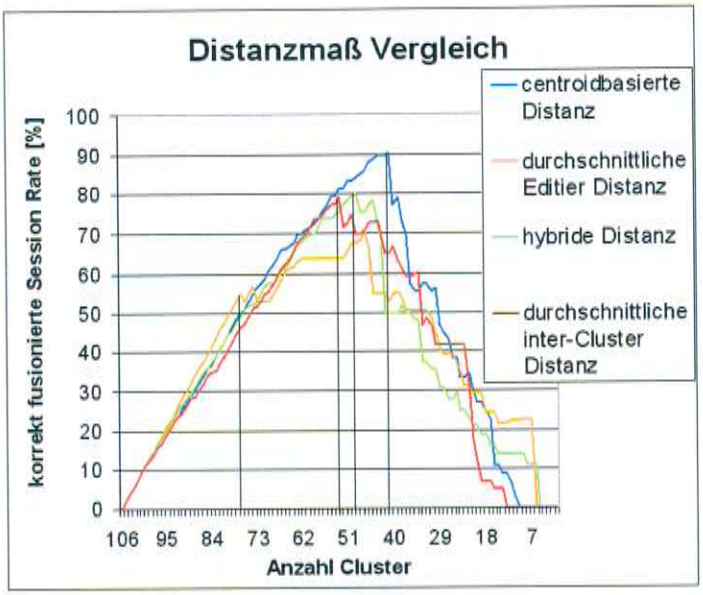

Abbildung 5.8: Distanzmaß Vergleich anhand der Rate der korrekt fusionierten Sessions für Set1

5.4 Mapping & Existenz-Dialoge

5.4.1 Mapping-Verfahren

Nachdem wir die Evaluation der Konfidenz bzw. des Clustering-Verfahrens mit den vorangegangenen Kapiteln weitgehend abgeschlossen haben, folgt nun die Evaluation der Wissensbasis-Bereinigung, was insbesondere die Evaluation der Label-Mapping-Verfahren beinhaltet. Für diese Evaluation macht es, anders als in der vorangegangenen Evaluation der Distanzmaße bzw. der Konfidenz, wenig Sinn das Clustering-Verfahren nur einmal auf dem gesamten Evaluationskorpus anzuwenden, denn dies entspräche nur einer Anwendung der Label-Mapping Verfahren. Viel mehr Sinn macht es hingegen, die Label-Mapping-Verfahren zu mehren Zeitpunkten anzuwenden, entspricht dies doch einem vorstellbaren Szenario in der praktischen Anwendung des Systems.

Die zugrundeliegende Idee wäre es den IslEnquirer beispielsweise jeden Abend, wenn die Lichter im Institut ausgegangen sind und kein Gesprächspartner mehr die Aufmerksamkeit des Systems bedarf, über das neu Gelernte „reflektieren“ zu lassen. „Reflektieren“ bedeutet in diesem Zusammenhang natürlich die Anwendung des Clusterings und der Label-

Mapping-Verfahren, wie wir sie in der vorliegenden Arbeit vorgestellt haben.

Um dieses Szenario auf unserem Evaluationssets (Set1 und Set2) nachzuempfinden, haben wir die Sets tageweise unterteilt. Ein so entstandener Set-Ausschnitt umfasst folglich jeweils die Sessions, die bis zu dem gegeben Zeitpunkt aufgezeichnet wurden. Set1 wurde auf diese Weise in 25 Ausschnitte zerlegt, Set2 in 24. Für jeden dieser Ausschnitte bzw. Reflektions-Zeitpunkte, wurde das Clustering-Verfahren und das anschließende Mapping-Verfahren separat durchgeführt. Die Ergebnisse eines Durchlaufes wurden dabei nicht für weitere Durchläufe berücksichtigt, so dass mögliche Fehlfusionen oder fehlerhafte Label-Mappings frühere Durchläufe keinen Einfluss auf spätere Ergebnisse haben konnten. Die Antworten zu gestellten Existenz-Dialog-Fragen vorheriger Durchläufe wurden jedoch gespeichert und kumulieren sich demnach im Laufe der Zeit.

Abbildungen 5.9, 5.10 und 5.11 zeigen die Ergebnisse dieser Evaluation für das erste Evaluationsset (Set1), wobei der Evaluationsindex, analog zu 5.1, der Anzahl betrachteter Sessions entspricht und Indizies mit Komastellen (z.B. 7.5) den Reflektions- bzw. Mapping-Zeitpunkten.

Die blau eingezeichneten Kurven mit dem Titel „noClustering" entsprechen in inverser Darstellung den Werten aus Abbildung 5.1 und fungieren als Referenzwerte. Die mit „offline" bezeichneten rötlichen Kurven entsprechen dem clusterbasierten Label-Mapping, wo bei das Clustering-Verfahren in dem einen Fall bei einem Konfidenzschwellwert von 95% und im anderen Fall bei einem von 90% unterbrochen wurde. Die vier gelblich eingezeichneten Kurven entsprechen dem dialogbasierten Label-Mapping, wobei sie sich im verwendeten Kondifenzschwellwert und der pro Durchlauf zugelassenen Anzahl an Existenz-Dialog-Fragen unterschieden. Die mit „_1Q" beschrifteten Kurven, gestatten eine Frage pro Durchlauf. Da eine beantwortet Existenz-Frage jedoch nicht genügt, um den Labelkonflikt eines Cluster aufzulösen, wurden zusätzlich dazu noch evaluiert welchen Einfluss es hat, dem Existenz-Dialog zu gestatten eine beliebe Anzahl von Fragen zu stellen, die nötig sind um den Labelkonflikt aufzulösen. In der Regel sind dazu zwei Fragen nötig, da die meisten Cluster nicht mehr als zwei verschiedene Label beinhalten. Die so evaluierten Existenz-Dialoge werden in den mit „_nQ" beschrifteten Kurven dargestellt.

Betrachtet man die Kurven genauer, fällt auf, dass sich die Kurven unterschiedlicher Konfidenzschwellwerte kaum von einander unterscheiden. So sind beispielsweise die beiden roten Kurven des clusterbasierten Label-Mapping bis auf wenige kurze Abweichungen weitgehend deckungsgleich. Dieser Umstand ist jedoch nicht weiter verwunderlich. Betrachtet man

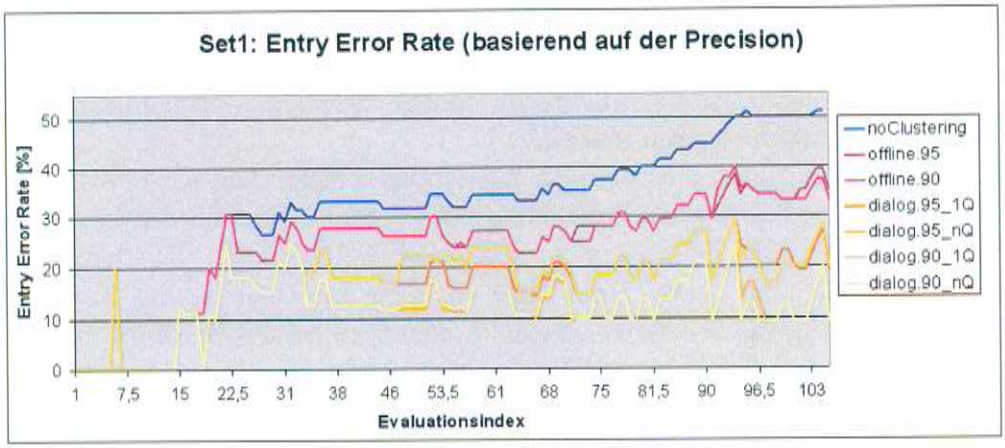

Abbildung 5.9: Entry Error Rate (EER) bei Set1

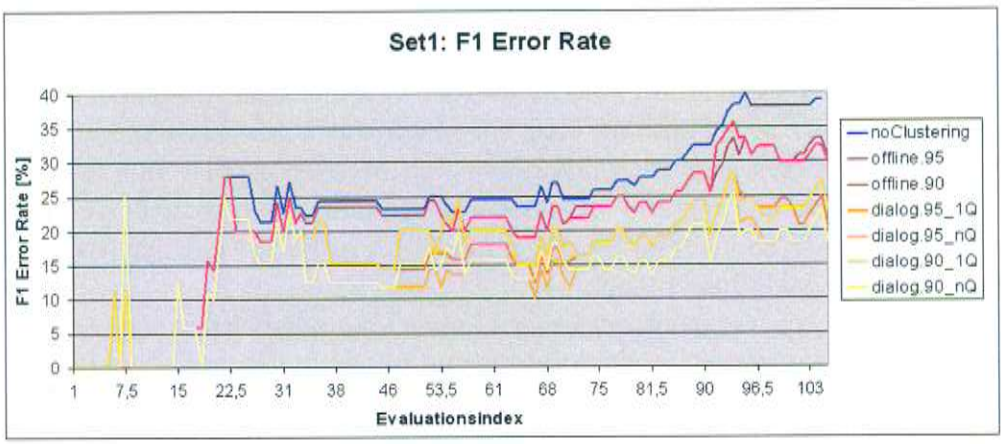

Abbildung 5.10: F-Measure Error Rate (F1ER) bei Set1

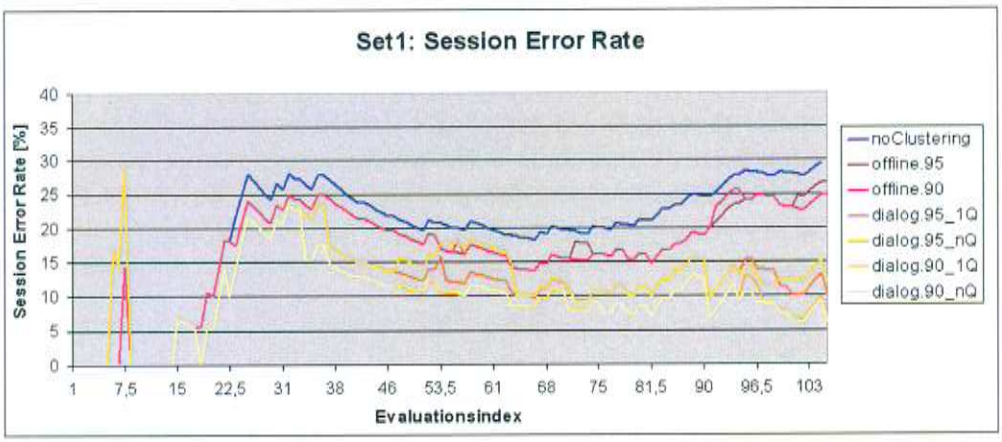

Abbildung 5.11: Session Error Rate (SER) bei Set1

den steilen Anstieg der Konfidenz in Abbildung 5.4, wird klar, dass eine Differenz von 5% im Konfidenzschwellwert bestenfalls eine oder zwei zusätzliche Cluster-Fusionen ausmacht, welche sich nicht zwingend auf das Label-Mapping auswirken müssen.

Weiter fallen die merkwürdigen Ausschläge einiger Kurven um den Indexwert 7,5 auf. Diese sind auf fehlerhafte Label-Mappings zurückzuführen, die durch eine Fehlfusion bei einem Schwellwert von 90% hervorgerufen wurden. Das diese Ausschläge so hoch ausfallen ist der Tatsache geschuldet, dass in der Wissensbasis in einem so frühen Stadium nur wenige Label bzw. Sessions vorliegen. Und dass es in diesem Stadium zu einer Fehlfusion kam, muss auch nicht weiter verwundern, denn die Konfidenz wurde, bedingt durch das in Kapitel 5.3.1 beschriebene Training, auf eine bestimmte Wissensbasis Größe ausgelegt, und weisst offensichtlich Schwächen auf, wenn diese Größe deutlich unterschritten wird.

Ganz allgemein lässt sich den Abbildungen 5.9 und 5.11 aber entnehmen, dass sich sowohl das clusterbasierte als auch das dialogbasierte Label-Mapping dazu eignen das Fehlerniveau der Wissensbasis zu senken. Wobei das F-Measure in Abbildung 5.10 vor allem durch die Verbesserung der Precision beeinflusst wird, wohingegen sich der Recall-Wert nur minimal verändert. Dieser Umstand ist jedoch verständlich, vergegenwärtigt man sich, dass die Wissensbasis-Bereinigung den Recall-Wert nur negativ nicht aber positiv beeinflussen kann.

Beim direkten Vergleich der beiden Label-Mapping-Verfahren zeigt sich, der Mehrwert zusätzlicher Dialog-Information, denn die dialogbasierte Label-Mapping-Methode übertrifft das clusterbasierten Label-Mappings in allen Belangen. Weniger deutlich, wenn auch immer noch gut erkennbar, ist der Unterschied zwischen den zwei Frage-Modi des Existenz-Dialogs. Mehr gestellte Fragen führt, wie man anhand der Abbildungen erkennen kann, zu besseren Ergebnissen, aber Umgekehrt lässt sich auch erkennen, dass bereits wenige Fragen eine signifikante Verbesserung gegenüber dem clusterbasierten Mapping erzielen. Eine genauere Analyse über Qualität und Quantität der gestellten Fragen findet sich in Kapitel 5.4.2.

Die Tabelle 5.6 fasst die Ergebnisse für das Set1 noch einmal zusammen. Insbesondere die Verbesserung der Wissensbasisqualität, durch das dialogbasierte Label-Mapping ist, mit Werten von 40% bis 80%, beeindruckend gut. Ausserdem fällt auf, dass das clusterbasierte Mapping-Verfahren bei Metriken die sich auf Label beziehen besser abschneidet, als bei Metriken die sich auf Sessions beziehen. Dies lässt sich insofern erklären, als dass Sessions einer Person nicht zwingend in einem Cluster vereint werden, so dass das clusterbaiserte Mapping sie nicht gleichermaßen mappt, wohinge-

Konf. Level	Evaluationsmetrik	Ergebnisse			Verbesserung	
		ohne	offline	dialog	offline	dialog
95% _1Q	Entry Error Rate	51.1	35.5	19.2	30.6%	62.4%
	F1 Error Rate	38.9	31.0	20.8	20.2%	46.6%
	Session Error Rate	29.5	26.7	10.0	9.7%	66.1%
90% _1Q	Entry Error Rate	51.1	33.3	20.0	34.8%	60.9%
	F1 Error Rate	38.9	29.8	23.1	23.3%	40.7%
	Session Error Rate	29.5	24.8	12.0	16.1%	59.2%
95% _nQ	Entry Error Rate	51.1	35.5	9.1	30.6%	82.2 %
	F1 Error Rate	38.9	31.0	18.4	20.2%	52.76%
	Session Error Rate	29.5	26.7	6.4	9.7%	78.3%
90% _nQ	Entry Error Rate	51.1	33.3	9.1	34.8%	82.2%
	F1 Error Rate	38.9	29.8	18.4	23.3%	52.8%
	Session Error Rate	29.5	24.7	6.2	16.1%	79.1%

Tabelle 5.6: Label-Mapping Ergebnisse für Set1

gen das dialogbasierte Mapping diese Cluster-Grenzen überwindet und die Sessions einheitlich mappen kann. Ein zweiter Grund sind die nicht auflösbaren Sessions in Set1, die proportional gesehen mit 48% einen größeren Anteil ausmachen als die auflösbaren Label mit 34%. Während das clusterbasierte Mapping diese Session überhaupt nicht auflösen kann, gibt es beim dialogbasierten Mapping zumindest die Möglichkeit solche Sessions mit eine 'Remove'-Markierung aus dem Korpus zu nehmen.

Da das erste Evaluationsset (Set1), einige Sessions enthält, die durch ein Label-Mapping nicht auflösbar sind, und dies vor allem Sessions von Gästen des Instituts betrifft, die nicht unbedingt in der Wissensbasis abgebildet werden müssen, wurde selbige Evaluation noch ein weiteres Mal auf einem zweiten Evaluationsset (Set2) durchgeführt. Die entsprechenden Ergebnisse werden analog zu den bisher vorgestellten Evaluationen in den Abbildungen 5.12, 5.13 und 5.14 gezeigt, wobei die Tabelle 5.7 die Ergebnisse zusammenfasst.

Da der Recall-Wert, wie man Abbildung 5.2 entnehmen kann, für dieses zweite Evaluationsset fast konstant auf 100% liegt und es keine nicht auflösbaren Sessions in dem Set gibt, sind die dort zu beobachtenden Evaluationsergebnisse sogar noch besser als bei dem ersten Evaluationsset. Dies gilt insbesondere für das F-Measure in Abbildung 5.13.

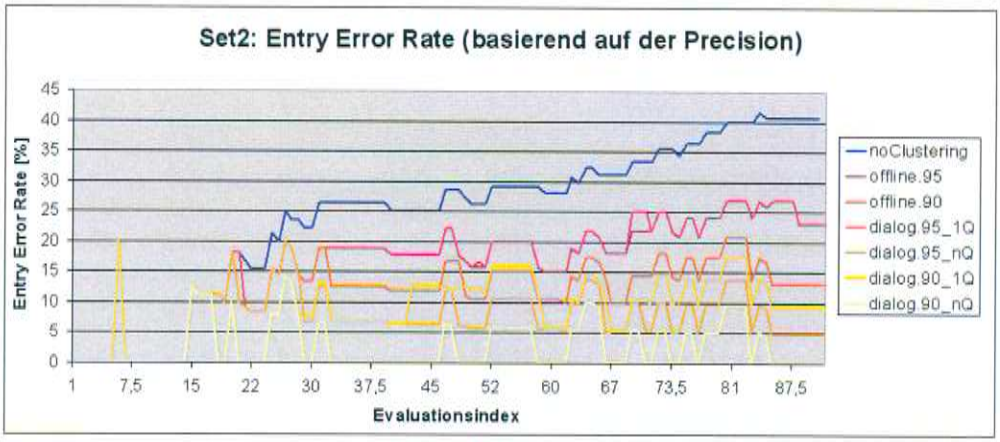

Abbildung 5.12: Entry Error Rate (EER) bei Set2

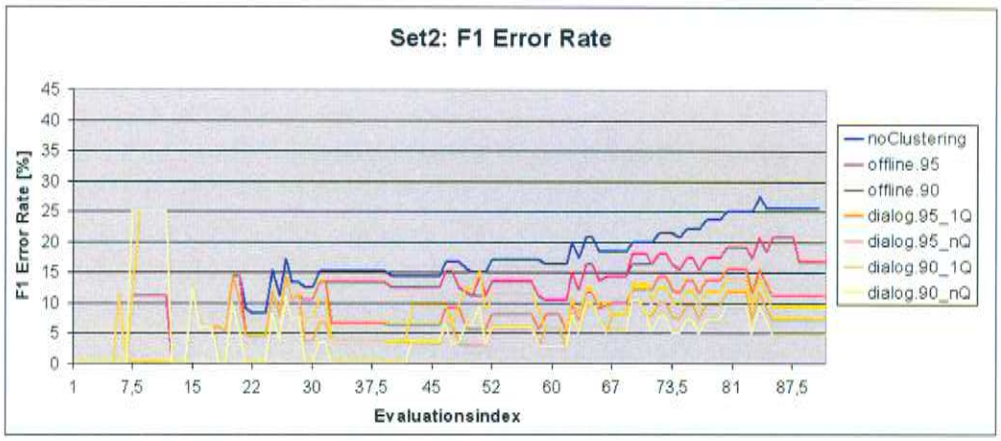

Abbildung 5.13: F-Measure Error Rate (F1ER) bei Set2

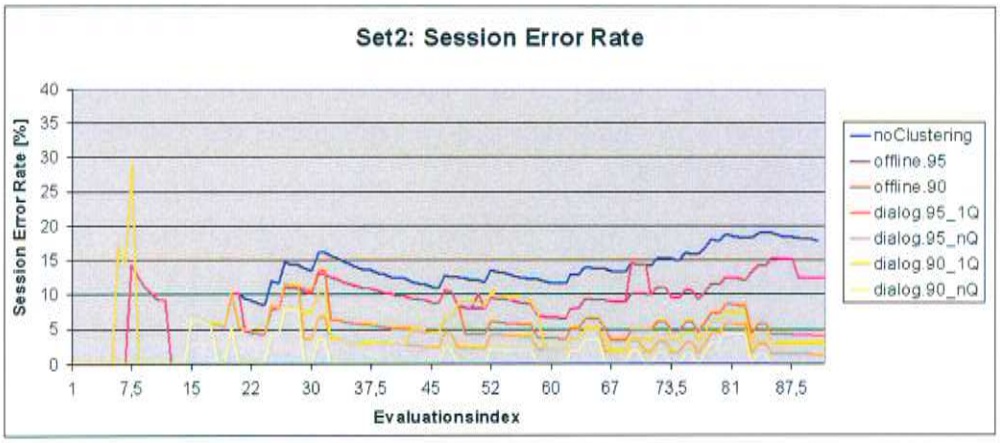

Abbildung 5.14: Session Error Rate (SER) bei Set2

Konf. Level	Evaluationsmetrik	Ergebnisse			Verbesserung	
		ohne	offline	dialog	offline	dialog
95% _1Q	Entry Error Rate	40.5	23.1	13.0	43.1%	67.8%
	F1 Error Rate	25.4	16.7	11.1	34.4%	56.3%
	Session Error Rate	17.8	12.2	3.9	31.3%	78.1%
90% _1Q	Entry Error Rate	40.5	23.1	9.1	43.1%	77.6%
	F1 Error Rate	25.4	16.7	9.1	34.4%	64.2%
	Session Error Rate	17.8	12.2	2.6	31.3%	85.2%
95% _nQ	Entry Error Rate	40.5	23.1	4.8	43.1%	88.3%
	F1 Error Rate	25.4	16.7	7.0	34.4%	72.5%
	Session Error Rate	17.8	12.2	1.3	31.3%	92.7%
90% _nQ	Entry Error Rate	40.5	23.1	0.0	43.1%	100%
	F1 Error Rate	25.4	16.7	4.8	34.4%	81.3%
	Session Error Rate	17.8	12.2	0.0	31.3%	100%

Tabelle 5.7: Label-Mapping Ergebnisse für Set2

5.4.2 Qualität und Quantität der gestellten Fragen

Wie wir den Evaluationen des vorangegangenen Kapitel entnehmen konnten, erzielt der dialogbasierte Mapping-Ansatz bessere Ergebnisse je mehr Fragen innerhalb eines Durchlaufes im Existenz-Dialog gefragt werden dürfen. Dieses Ergebnis ist nicht weiter verwunderlich, stehen diesen doch auch mehr Informationen zur Verfügung. Da der Dialog mit einer trusted-Person jedoch eine wertvolle Resource ist, interessieren wir uns neben der Verbesserung der Wissensbasis auch dafür, wie effizient die Zeit der trusted-Person durch den Dialog genutzt wurde. Oder um es anders auszudrücken, wie effektiv die gestellten Fragen bzw. wie gut die Sortierung nach Priorität der zu erfragenden Label war.

Zu diesem Zweck berechnen wir den Nutzen pro Frage, in dem die Anzahl gestellter Fragen der Anzahl korrekt aufgelöster bzw. korrekterweise mit 'Remove'-markierter Sessions gegenüberstellen. Dabei ist zu beachten, dass die Benefit-Rate auch Werte über 100% annehmen kann, wenn beispielsweise ein gefragtes Label mehr als nur eine Session korrigiert.

$$\text{Benefit-Rate} = \frac{\text{Anzahl korrigierter Sessions}}{\text{Anzahl gestellter Fragen}} \tag{5.7}$$

Tabelle 5.8 zeigt die Werte der Benefit-Rate für die im letzten Kapitel vorgestellten Evaluationen. Sie niedrigeren Benefit-Werte für das zweite

	Benefit-Rate			
	dialog.95_1Q	*dialog.95_nQ*	*dialog.90_1Q*	*dialog.90_nQ*
Set1	92.0%	74.3%	84.0%	70.3%
Set2	54.4%	46.9%	58.3%	48.5%

Tabelle 5.8: Benefit-Rate für Set1 und Set2

Evaluationsset (Set2) lassen sich mit der deutlich geringeren Anzahl fehlerhaft gelabelter Sessions erklären. Für beide Evalautionssets ist hingegen zu erkennen, dass die Benefit-Rate mit zunehmender Anzahl verfügbarer Fragen abnimmt.

Etwas genau zeigt der linke Teil der Abbildung 5.15 die Benefit-Rate für das vollständige Set1 im zeitlichen Verlauf und um die Verzerrung durch die stetig ansteigende Anzahl von Fragen aufzuheben zeigt der rechte Teil der Abbildung die Session Korrekturen noch einmal in absoluten Zahlen.

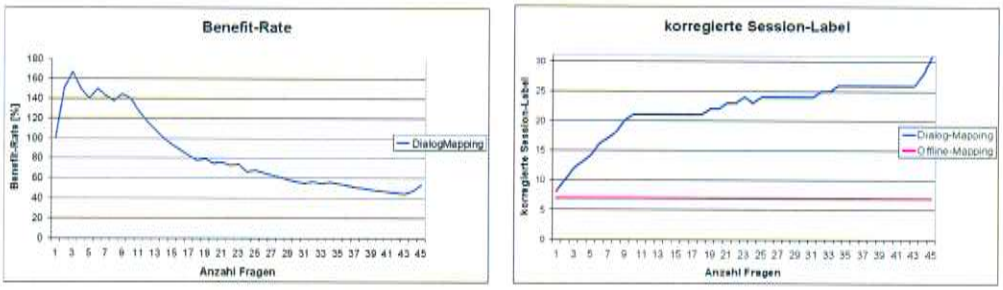

Abbildung 5.15: Benefit-Rate pro Frage auf Set1

Dem Kurvenverlauf können wir entnehmen, dass die Sortierung der zu erfragenden Label nach Priorität gut gelingt, wobei der Anstieg im letzten Teil der Grafiken eine Ausnahme bildet. Bei den dort korrigierten Labeln, handelt es sich ausschließlich um mehrfach bestätigte Fehllabel des zweiten Typus. Da die Label in mehreren Sessions bestätigt wurden, führt die Sortierung nach $rank_{\text{Häufigkeit}}$ (siehe Gleichung 4.4) zu einer Fehleinschätzung. Und da diese Sessions auch noch ohne ein anderes Label zusammen geclustert wurden und daher keinen Labelkonflikt verursachen, verstärken die Sortierungen nach $rank_{\text{Unsicherheit}}$ und $rank_{\text{Verwechslungsgrad}}$ diese Fehleinschätzung noch weiter.

6 Diskussion & Ausblick

6.1 Zusammenfassung und Diskussion der Ergebnisse

Der vorgestellte Ansatz zeigt, dass eine Wissensbasis-Bereinigung auf Basis von Clustering und Label-Mapping möglich ist und dass durch ein proaktives Verhalten des Robotors und der Verwendung einfacher Existenzfragen die Korrekturrate der Wissensbasis noch weiter gesteigert werden kann.

Der hierzu vorgeschlagene Clustering-Ansatz geht dabei iterativ agglomerativ vor, wobei eine auf inter- und intra-Cluster-Distanzen basierende Konfidenz sowohl als adaptives Distanzmaß für die Fusion der Session-Cluster genutzt wird, als auch mit Hilfe eines zuvor definierten Schwellwerts als Abbruchbedinung.

Für die Konfindezbestimmung wurden dabei verschiedene zugrundeliegende Distanzmaße und Konfidenzmerkmale evaluiert, wobei sich die Kombination aus centroidebasierter inter-Cluster-Distanz und durch Session Anzahl approximierte intra-Cluster-Distanz als besonders gut erwiesen hat.

Die Diskrepanz zwischen 78.3% Fehllabel-Detektions-Rate des Clustering-Verfahrens (siehe Abbildung 5.4) und der Verbesserung der Precision von 30.6% (siehe Tabelle 5.6) zeigt die Schwierigkeit, nur auf Basis der geclustereten Sessions, die fehlerhaften Label korrekt aufzulösen.

Das erste Evaluationsset (Set1) verzerrt diese Zahlen auch noch durch die Existenz nicht auflösbarer Sessions, was eine der Schwächen des Ansatzes zeigt. Durch die Koppelung zwischen Clustering und Labelauflösung lassen sich fehlerhafte Label nur dann auflösen, wenn zumindest einige der Sessions der betroffenen Person korrekt in der Wissensbasis vorliegen, andernfalls lässt sich eine derart fehlgelabelte Session bestenfalls detektieren. Eine solche Detektion ist jedoch nur dann möglich wenn A) weit gehend ausgeschlossen werden kann das Fehlfusionen während des Clusterings entstehen, was die von uns untersuchte Konfidenz sicherstellt, und B) die betroffenen Sessions nach dem Clustering-Verfahren nicht als ein-

zele Sessions in einem Cluster verbleiben. Die korrekt-fusionierte-Session-Rate, in Abbildung 5.4, von 90%, weist eine gute Abdeckung der Sessions auf, zeigt aber auch Spielraum nach oben.

Bei der Evaluationen auf dem zweiten Evaluationsset (Set2) konnte gezeigt werden, dass sich ohne die Verzerrung durch nicht auflösbare Session-Label deutlich bessere Ergebnisse erzielen lassen. Dennoch weist das clusterbasierte Label-Mapping-Verfahren mit einer Verbesserung der Precision von 43.1% immer noch eine Diskrepanz zur Fehllabel-Detektions-Rate auf mit 80%. Dies liegt mit unter auch daran, dass Sessions einer Person nicht zwingend alle in einem Cluster vereint werden, so dass beim Mapping möglicherweise nicht alle Sessions gleichermaßen gemappt werden und einzele Fehler in der Wissensbasis verbleiben.

Dieses Problem wird erst durch das dialogbasierten Mapping gelöst, welches auch über Cluster-Grenzen hinweg Label auflösen kann, was unter anderem auch die weit besseren Ergebnisse dieses Label-Mapping-Verfahrens erklärt. Ein weiterer Vorteil des dialogbasierten Mappings ist die Möglichkeit fragliche Session, deren Label die nicht aufgelöst werden können, mit einer „Remove“ Markierung aus dem Korpus zu nehmen. Auf diese Weise erzielt das dialogbasierte Mapping-Verfahren, je nach Konfidenzschwellwert und Anzahl zulässiger Existenz-Dialog-Fragen, zwischen 60% und 82% relativer Precision Verbesserung, auf dem ersten (Set1), und zwischen 67% und 100% auf dem zweiten Evaluationsset (Set2). Diese Zahlen zeigen deutlich welchen Mehrwert die Dialoginformation für die Wissensbasis-Bereinigung bietet.

Ähnlich dem Vorgehen beim Aktive Learning werden auch bei dem von uns vorgeschlagenen Ansatz die zu erfragenden Label nach Priorität sortiert. Die Analyse dieser Sortierung hat gezeigt, dass dies, bis auf die Ausnahme mehrfach fehlbestätigter Label, gut gelingt.

6.2 Ausblick

Für die Zukunft sind folgende Erweiterungen des vorgestellten Ansatzes denkbar. Zunächst gilt es zu untersuchen inwiefern sich das von uns dargestellte iterativ agglomerative Clustering Verfahren, durch die Verwendung von Cluster Feature (CF)[1] bezüglich des zeitlichen Aufwands beschleunigen und somit parallel zu den Dialogen ausführen lässt. Außerdem wäre die Untersuchung weiterer, weniger durch Beleuchtungsverhältnisse beeinflusste, Merkmale denkbar, wobei auch eine Dimensionsreduktion auf

[1] siehe Kapitel 3.1.2.1

den DCT-Vektoren eine weitergehende Analyse wert ist, verspricht dies doch den Clustering-Ansatz noch weiter zu optimieren. Sicher auch interessant für eine weitergehende Analyse wäre die Nutzung der in den Dialogen gesammelten Informationen. Diese ließen sich nicht nur für die Verbesserung des Clusterings nutzen, sondern auch für die Optimierung der Session-Label-Mappings, denn diese berücksichtigen derzeit nicht den Informationsverlust, der mit den „Remove"-Markierungen nicht auflösbarer Sessions einhergeht. Auch interessant für die Mapping Optimierung, könnte die Untersuchung der Cluster-Ergebnisse über mehrere Durchläufe hinweg sein, da dies eine Konfidenzbildung über die Clustering Ergebnisse erlauben würde, was zu Detektion von Fehlfusionen genutzt werden könnte.

Literaturverzeichnis

[Car92] CARPENTER, BOB: *The logic of typed feature structures.* Cambride Univerity Press, 1992.

[CCS00] CHAPELLE, O., N. CRISTIANINI und A. SMOLA: *Query learning with large margin classifiers.* Proc. 17th International Conference on Machine Learning, Seiten pp. 111–118, 2000.

[EKSX96] ESTER, M., H.-P. KRIEGEL, J. SANDER und X. XU: *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.* Proc. Int. Conf. on Knowledge Discovery and Data Mining, 1996.

[Faw03] FAWCETT, TOM: *ROC Graphs: Notes and Practical Considerations for Data Mining Researchers.* Technischer Bericht, Tech Report HPL-2003-4, HP Laboratories., 2003.

[FSST97] FREUND, Y., H. S. SEUNG, E. SHAMIR und N. TISHBY: *Selectiv sampling using the query by committee algorithm.* Machine Learning, 28, 1997.

[GHW08] GROSSE, PHILIPP, HARTWIG HOLZAPFEL und ALEX WAIBEL: *Confidence Based Multimodal Fusion for Person Identification.* ACM Multimedia, 2008.

[GRS98] GUHAA, SUDIPTO, RAJEEV RASTOGIB und KYUSEOK SHIM: *Cure: an efficient clustering algorithm for large databases.* Proc. ACM SIGMOD Conf. Management of Data, Seiten pp. 73–84., 1998.

[GRS99] GUHA, SUDIPTO, RAJEEV RASTOGI und KYUSEOK SHIM: *ROCK: A Robust Clustering Algorithm for Categorical Attributes.* Proceedings of the 15th International Conference on Data Engineering, 1999.

[HK06] HAN, JIAWEI und MICHELINE KAMBER: *Data Mining, Concepts and Techniques.* Morgan Kaufmann, 2006.

[Hol08] HOLZAPFEL, HARTWIG: *A Dialogue Manager for Multimodal Human-Robot Interaction and Learning of a Humanoid Robot.* Industrie Robots Journal 35-6, 2008.

[Hol09] HOLZAPFEL, HARTWIG: *Acquiring and Maintaining Knowledge by Natural Multimodal Dialog.* Doktorarbeit, Universität Karlsruhe (TH), 2009.

[HS95] HERNANDEZ, M. A. und S. J. STOLFO: *The merge/purge problem for large databases.* In Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, Seiten pages 127–138, 1995.

[HS98] HERNANDEZ, M. A. und S. J. STOLFO: *Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem.* Journal of Data Mining and Knowledge Discovery, vol. 2:pp. 9–37., 1998.

[HW07] HOLZAPFEL, HARTWIG und ALEX WAIBEL: *Behavior Models for Learning and Receptionist Dialogs.* Proceedings of Interspeech, 2007.

[KHK99] KARYPIS, G., E.-H. HAN und V. KUMAR: *CHAMELEON: A Hierarchical Clustering Algorithm using dynamic modeling.* Computer, Volume: 32, Issue: 8:pp. 68–75, 1999.

[LG94] LEWIS, D. D. und W. A. GALE: *A sequential algorithm for training test classifiers.* Proc. of SIGIR-94 17th ACM International Conference on Research and Development in Information Retrieval, Seiten pp. 3–12, 1994.

[LRZ09] *http://www.lrz-muenchen.de/~wlm/ilm_l11.htm.* Stand: März 2009.

[MAU94] MOSES, Y., Y. ADINI und S. ULLMAN: *Face Recognition: The Problem of Compensating for Changes in Illumination Direction.* European Conf. Computer Vision,, Seiten pp. 286–296, 1994.

[Put08] PUTZE, FELIX: *Social User Model Acquisition through Network Analysis and Interactive Learning.* Diplomarbeit, Universität Karlsruhe (TH), 2008.

[RQD00] Reynolds, Douglas A., Thomas F. Quatieri und Robert B. Dunn: *Speaker verification using adapted gaussian mixture models.* Digital Signal Processing, 10:19–41, 2000.

[SB02] Sarawagi, Sunita und Anuradha Bhamidipaty: *Interactive Deduplication using Active Learning.* In Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002), 2002.

[SC00] Schohn, G. und D. Cohn: *Less is more: Active learning with support vektor machines.* Proc. 17th International Conference on Machine Learning, Seiten pp. 839–846, 2000.

[TC01] Tong, S. und E. Chang: *Support vektor machine active learning for image retrieval.* ACM int. conf. on Multimedia, Seiten pp. 107–118, 2001.

[TK01] Tong, S. und D. Koller: *Support vektor machine active learning with applications to text classification.* Journal of Machine Learning Research, 2:45–66, 2001.

[VJ03] Viola, Paul und Michael J. Jones: *Fast Multi-view Face Detection.* Technischer Bericht, Technical Report TR2003-96, Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, Juni 2003.

[Wik09] *http://de.wikipedia.org/wiki/Diskrete_Kosinustransformation.* Stand: März 2009.

[Zig09] *http://www.cs.cf.ac.uk/Dave/Multimedia/node238.html.* Stand: März 2009.

[ZRL96] Zhang, Tian, Raghu Ramakrishnan und Miron Livny: *BIRCH: An Efficient Data Clustering Method for Very Large Databases.* Proceedings of the 1996 ACM SIGMOD, 1996.