HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

ins INSTITUTE FOR
NETWORKED SOLUTIONS

Text Mining

Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2013

Autor:	Quentin Willimann
Betreuer:	Prof. Hansjörg Huser
Projektpartner:	INS
Experte:	Stefan Zettel
Gegenleser:	Prof. Oliver Augenstein

Erklaerung

Ich, Quentin Willimann, erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.
- dass ich keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt habe.

Ort, Datum:

Name, Unterschrift:

Textmining

Aufgabenstellung für Quentin Willimann

Aufgabe:
Text Mining wendet Methoden des Data Mining auf unstrukturierte Texte an. Das „Mining" in den Textbeständen hat die Aufgabe, nicht-triviales und verwertbares Wissen, sowie Beziehungen zwischen Texten und Textfragmenten zu entdecken.
Ziel dieser Arbeit ist das tutorial-mässige Erarbeiten der verschiedenen Prozessschritte im Text-Mining Prozess sowie das Anwenden der Ergebnisse in Text Klassifikation und im automatischen Generieren von Zusammenfassungen.

Folgende Prozessschritte sollen detailliert analysiert werden:

- Datenaufbereitung und -verarbeitung (Bereitstellung des Dokumentenkorpus, Tokenizing, syntaktische und semantische Analysen)
- Modellbildung für Aufgaben wie Summarization, Informationsextraktion, Klassifikation, Clustering.
- Darstellung bzw. Validierung der Resultate.

Dabei sollen theoretische Konzepte (u.a. Similarity, Distance) und Begriffe sowie praktische Aspekte (Tooling mit Beispieldateien) dokumentiert werden.

Resultate:

- Literaturrecherche
- Prozess für die Klassifikation und für die automatische Zusammenfassung
- Dokumentation der Arbeiten

Projektteam

Quentin Willimann

Betreuung HSR

Hansjörg Huser

Experte

Stefan Zettel

Projektabwicklung

Termine:

- Beginn der Arbeit: M**o., 16. Sept. 2013**
- Zwischenpräsentation für Gegenleser und evtl Experte: ca. anfangs Nov.
- Abgabetermin Abstract für DA-Broschüre: **12.Dez.2013**
- Abgabetermin Kurzfassung/Poster/Mgmt-Summary zum Review: 16. Dez. 2013
- Abgabetermin (ink. Poster): **20. Dez. 2013, 12.00 Uhr**
- Mündliche BA-Prüfung : **16. Jan 2014, 15.30 Uhr**

Arbeitsaufwand

Für die erfolgreich abgeschlossene Arbeit werden 12 ECTS angerechnet. Dies entspricht einer Arbeitsleistung von mind. 360 Stunden pro Student.

Hinweise für die Gliederung und Abwicklung des Projektes:

Gliedern Sie Ihre Arbeit in 4 bis 5 Teilschritte. Schliessen Sie jeden Teilschritt mit einem Meilenstein ab. Definieren Sie für jeden Meilenstein, welche Resultate dann vorliegen müssen!

Folgende Teilschritte bzw. Meilensteine sollten Sie in Ihrer Planung vorsehen:

Schritt 1: Projektauftrag inkl. Projektplan (mit Meilensteinen),

- Meilenstein 1: Review des Projektauftrages abgeschlossen. Projektauftrag von Auftraggeber und Dozent genehmigt
- Entwickeln Sie Ihre Algorithmen in einem iterativen, inkrementellen Prozess: Planen Sie möglichst früh einen ersten lauffähigen Prototypen mit den wichtigsten und kritischsten Kernfunktionen. In die folgenden Phasen können Sie dieses Kernsystem schrittweise ausbauen und testen.
- Falls Sie in Ihrer Arbeit neue oder Ihnen unbekannte Technologien einsetzen, sollten Sie parallel zum Erarbeiten des Projektauftrages mit dem Technologiestudium beginnen.
- Verwalten Sie ihre Software und Dokumente auf einem geeigneten Repository. Stellen Sie sicher, dass der/die Betreuer jederzeit Zugriff auf das Repository haben und dass das Projekt anhand des Repositories jederzeit wiederhergestellt werden kann.

Projektadministration

- Führen Sie ein individuelles Projekttagebuch aus dem ersichtlich wird, welche Arbeiten Sie durchgeführt haben (inkl. Zeitaufwand). Diese Angaben sollten u.a. eine individuelle Beurteilung ermöglichen.
- Dokumentieren Sie Ihre Arbeiten laufend. Legen Sie Ihre Projektdokumentation mit der aktuellen Planung und den Beschreibungen der Arbeitsresultate elektronisch im Repository ab. Dieser Projektordner sollte jederzeit einsehbar sein.

Inhalt der Dokumentation

Bei der Abgabe muss jede Arbeit folgende Inhalte haben:

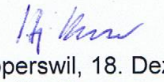
- Dokumente gemäss Vorgabe: https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html
- Aufgabenstellung
- Technischer Bericht
- Projektdokumentation
- Die Abgabe ist so zu gliedern, dass die obigen Inhalte klar erkenntlich und auffindbar sind.
- Zitate sind zu kennzeichnen, die Quelle ist anzugeben.
- Verwendete Dokumente und Literatur sind in einem Literaturverzeichnis aufzuführen.

Fortschrittsbesprechung:

Regelmässig findet zu einem fixen Zeitpunkt eine Fortschrittsbesprechung statt.
Teilnehmer: Dozent und Studenten, bei Bedarf auch Vertreter der Auftraggeber
Termin: jeweils Mittwoch, 14h, Raum 6.010 (Abweichungen werden rechtzeitig kommuniziert)
Traktanden

- Was wurde erreicht, was ist geplant, welche Probleme stehen an

Falls notwendig, können weitere Besprechungen / Diskussionen einberufen werden.



Rapperswil, 18. Dez. 2013
Hansjörg Huser

HOCHSCHULE FÜR TECHNIK RAPPERSWIL

Abstract

Institute for network solutions

Textmining

von Quentin Willimann

Text Mining hat die Aufgabe, nicht-triviales und verwertbares Wissen, sowie Beziehungen zwischen Texten und Textfragmenten zu entdecken. Mehrere Fachrichtungen sind dabei zu berücksichtigen: Informations Retrieval, Data Mining, Maschinelled Lernen, Statistik und Computerlinguistik. Diese Begriffe werden erklärt und deren Zusammenhang mit Text Mining wird aufgezeigt. In dieser Bacherlorarbeit werden anhand von Klassifizierungsproblemen und der automatischen Generierung von Zusammenfassung die Prozessschritte eruiert.
Folgende Schritte werden erarbeitet und dokumentiert:

1. Preprocessing (Tokenisierung, Stemming, Vector-Space-Modelling)
2. Postprocessing (Dimension Reduktion)
3. Clustering (K-Means)
4. Classification
5. Evaluation (Cosine Similarity, Accuracy)

Nebst den herkömmlichen Algorithmen werden die Resultate mittels linguistischen Methoden verbessert, wie POS Tagging, Synonyme und Hyperonyme.
Diese Bachelorarbeit beschreibt einen Algorithmus zur Klassifizierung von Texten und einen zur Generierung von Zusammenfassungen.
Mit dem finalen Klassifikationsalgorithmus wird auf dem verwendeten Datensatz „newsgroups“ eine Exaktheit von rund 92.5% für zwei Klassen, 91.3% für drei und 83.5% für vier erreicht. Problematisch sind vor allem Klassen, die nahen zueinander liegen und sich je nachdem sogar überschneiden.
Bei der automatischen Zusammenfassung liegt das Augenmerk beim Herausfinden der Themen, welche ein Dokument umfasst. Aus einem wissenschaftlichen Text werden Sätze selektiert, welche den Text möglichst gut zusammenfassen sollen. Diese extraktive Zusammenfassung wird dann mit Hilfe von Cosine Similarity mit dem verfügbaren Abstract verglichen. Dabei wird ein Bestwert von 0.7 erreicht.

Management Summary

Augangslage
Text Mining hat die Aufgabe interessantes und nicht triviales Wissen aus unstrukturierten, bzw. schwach-strukturierten Texten zu extrahieren. Für den Erfolg dieser Aufgabe müssen mehrere Fachrichtungen in Betracht gezogen werden, Informations Retrieval, Data Mining, Maschinelles Lernen, Statistik und Computerlinguistik. In dieser Bachelorarbeit sollen anhand von zwei Teilproblemen (Klassifikation und automatische Zusammenfassung) die Prozessschritte von Textmining eruiert werden.

Vorgehen/Technologien
Für diese Bachelorarbeit werden zwei verschiedene Tools eingesetzt. Zum einen Rapidminer für die Entwicklung eines Klassifizierungsmodells und zum anderen Python für das gezielte Lernen und Bearbeiten der automatischen Zusammenfassung. Bei beiden Teilaufgaben geht es in erster Linie darum den ganzen Ablauf zu modellieren. Danach wird versucht iterativ die Resultate zu optimieren.
Die erarbeiteten Schritte sind unter anderem folgende:

1. Datenaufbereitung
2. Datennachbearbeitung, Reduktion der Daten
3. Gruppierung der Dokumente
4. Klassifizierung
5. Evaluation

Für die Klassifikation werden mehrere Verfahren (Support Vektor Maschine, Naive Bayes, ID3,...) angewandt und miteinander verglichen.
Nebst den herkömmlichen Algorithmen wird auch versucht, die Resultate mittel linguistischen Methoden zu verbessern. Dabei werden Prozesse verwendet, welche die Wörter in Typen unterscheidet (Substantive, Verben, Adjektive...) oder die Wörter als Synonyme erkennt und zusammenfasst.

Ergebnis

Klassifikation: Da die gegebenen Dokumente eher kurz sind, ist eine genaue Vorhersage, zu welcher Klasse ein Text gehört nicht ganz einfach. Nach dem Lernen des Klassifikationsmodells anhand 1000 Texten pro Klasse wird versucht eine kleinere Menge an Dokumenten vorherzusagen. Dabei erreichte meine Prozesskette eine Genauigkeit von rund 92.5% für zwei Klassen, 91.3% für drei Klassen und 83.5% für vier Klassen. Problematisch sind vor allem Klassen, die nahen zueinander liegen und sich je nachdem sogar überschneiden.

Automatische Zusammenfassung: Auch bei dieser Problemstellung sind wieder die im Vorfeld erarbeiteten Prozessschritte involviert. Jedoch liegt hier das Augenmerk mehr beim Herausfinden der Themen, welche ein Dokument umfasst. Ein Erlernen und Clustern basiert hier auf noch kürzeren Abschnitten. Aus einem wissenschaftlichen Text werden Sätze selektiert, welche den Text möglichst gut zusammenfassen sollen. Dieser generierte Text wird dann mit Hilfe von Cosine Similarity mit dem verfügbaren Abstract verglichen. Es wird eine Ähnlichtkeit mit dem Abstract von rund 70% erreicht.

Danksagung

Ich möchte mich an dieser Stelle bei all denjenigen bedanken, die mich während dieser Bachelorarbeit unterstützt und motiviert haben.

Ganz besonders möchte ich mich bei Herrn Prof. Huser, der meine Arbeit und somit auch mich betreut hat, bedanken. Durch seine Offenheit gegenüber diesem Thema hat er mich dazu gebracht, über meine Grenzen hinaus zu denken.

Des Weiteren gebührt mein Dank Herrn Willimann für die moralische Unterstützung und die zahlreichen Stunden, die er Korrektur gelesen hat. Er wies mich auf Schwächen hin und zeigte auf, wo noch Erklärungsbedarf bestand.

Allgemein möchte ich meiner Familie danken für die herzliche Unterstützung während des ganzen Studiums.

Inhaltsverzeichnis

Erklärung i

Aufgabenstellung ii

Abstract iv

Management Summary vi

Danksagung viii

Inhaltsverzeichnis ix

Abbildverzeichnis xii

Tabellenverzeichnis xiii

Abkürzungen xiv

1 Einleitung 1

2 Definition: Text Mining 3
- 2.1 Data Mining . . . 3
- 2.2 Information Retrieval . . . 4
  - 2.2.1 Relevanz . . . 4
- 2.3 Maschinelles Lernen . . . 5
- 2.4 Computer Linguistik . . . 5

3 Grundlagen 6
- 3.1 Preprocessing . . . 6
  - 3.1.1 Tokenization . . . 7
  - 3.1.2 Filterung von Stopwords . . . 7
  - 3.1.3 Gross/Kleinschreibung Ignorieren . . . 8
  - 3.1.4 Token Filterung (Wortlänge) . . . 8
  - 3.1.5 POS Tagging . . . 8

3.1.6 Stemming . . . 9
3.1.7 WordNet Integration . . . 9
3.1.8 N-Grams . . . 9
3.1.9 Vektorraum-Modell . . . 10
3.2 Postprocessing . . . 11
3.2.1 χ^2 Statistik . . . 12
3.2.2 Attribut Reduzierung . . . 12
3.3 Clustering Algorithmen . . . 13
3.3.1 K-Means . . . 13
3.4 Klassifikation . . . 14
3.4.1 Support Vector Machine . . . 14
3.4.2 Naïve Bayes . . . 15
3.4.3 ID3/Decision Tree . . . 15
3.5 Evaluation . . . 16
3.5.1 Relevanzbewertung . . . 16
3.5.2 Similarity Bewertung . . . 18
3.5.3 Cross-Validation . . . 18

4 Tools **19**
4.1 Rapidminer . . . 19
4.2 Python . . . 20
4.3 NLTK - Natural Language Toolkit . . . 20

5 Datensatz **22**
5.1 Classification . . . 22
5.2 Automische Zusammenfassung . . . 23

6 Umsetzung und Resultate **24**
6.1 Klassifikation I . . . 24
6.1.1 Preprocessing I . . . 24
6.1.1.1 Prozess I . . . 25
6.1.1.2 Prozess II . . . 25
6.1.1.3 Prozess III . . . 25
6.1.1.4 Prozess IV . . . 26
6.1.1.5 Prozess V . . . 26
6.1.1.6 Fazit . . . 26
6.1.2 Preprocessing II . . . 28
6.1.2.1 Prozess VI . . . 28
6.1.2.2 Prozess VII . . . 28
6.1.2.3 Prozess VIII . . . 28
6.1.2.4 Prozess IX . . . 29
6.1.2.5 Fazit . . . 29
6.1.3 Vergleich von Klassifikationsalgorithmen . . . 30
6.1.3.1 Fazit . . . 30

6.1.4 Weitere Tests . . . 31
6.1.5 Klassifikation: Grosse Datenmenge bestimmt Kleine . . . 32
6.1.5.1 Resultat . . . 32
6.2 Klassifikation II . . . 32
6.2.1 Einstellungen . . . 32
6.2.1.1 Daten . . . 33
6.2.1.2 Preprocessing . . . 33
6.2.1.3 Postprocessing . . . 33
6.2.2 Resultate . . . 33
6.2.3 Fazit . . . 34
6.3 Klassifikation III . . . 34
6.3.1 Prozess . . . 34
6.3.2 Resultate . . . 34
6.3.3 Fazit . . . 36
6.4 Automatische Zusammenfassung . . . 37
6.4.1 Systemevaluation . . . 37
6.4.2 Prozess . . . 37
6.4.3 Resultate . . . 39
6.4.4 Fazit . . . 40

7 Schlussfolgerungen **41**
7.1 Ausblick . . . 45

8 Persönlicher Bericht **46**

Literaturverzeichnis **48**

Abbildungsverzeichnis

2.1 Pipelinemodell der Sprachverarbeitung[1] 5

4.1 Beispiel von Operatoren in Rapidminer. 20

5.1 Beispiel eines Dokumentes aus der Klasse comp.graphics. 23

6.1 Preprocessing I: Veränderung der Vorhersage durch Hinzufügen von Teilschritten 27
6.2 Preprocessing II: Vergleich von linguistischen Ansätzen 30
6.3 Klassifikation: Optimierungsversuche 31
6.4 Finaler Klassifizierungsprozess 35
6.5 Automatische Zusammenfassung: Prozess 37
6.6 Beispiel einer generierten Zusammenfassung des Textes [2] 39

Tabellenverzeichnis

3.1 Definition von tp, fp, fn und tn. 16

5.1 Datensätze 22

6.1 Klassifikation I, Preprocessing I, Datensatz 24
6.2 Klassifikation I, Prozess I 25
6.3 Klassifikation I, Prozess II 25
6.4 Klassifikation I, Prozess III 26
6.5 Klassifikation I, Prozess IV 26
6.6 Klassifikation I, Prozess V 27
6.7 Klassifikation I, Prozess VI 28
6.8 Klassifikation I, Prozess VII 29
6.9 Klassifikation I, Prozess VIII 29
6.10 Klassifikation I, Prozess IX 29
6.11 Klassifikation II, Daten 33
6.12 Klassifikation II, Resultate 33
6.13 Klassifikation III, Daten 34
6.14 Klassifikation III: Resultate, 2 Klassen 36
6.15 Klassifikation III: Resultate, 3 Klassen 36
6.16 Klassifikation III: Resultate, 4 Klassen 36
6.17 Automatische Zusammenfassung: Resultate 40

Abkürzungen

POS	**P**art **O**f **S**peech
NER	**N**amed **E**ntity **R**ecognition
KDD	**K**nowledge **D**iscovery in **D**atabases
KDT	**K**nowledge **D**iscovery in **T**extual Databases
SynSet	**Syn**onym **Set**

Kapitel 1

Einleitung

Auf Grund des Anstiegs an verfügbaren Daten fällt es einem Benutzer immer schwerer, die für ihn relevanten Informationen zu finden. Um diesen ganzen Informationsfluss zu bündeln und zu filtern werden Text Mining Algorithmen eingesetzt, die dem User bei der Auffindung seines Materials unterstützen sollen.
Diese Arbeit wird dem Leser aufzeigen, wie solche Text Mining Systeme aufgebaut werden, dabei liegt der Fokus auf der Klassifikation von Texten und darauf aufbauend das automatische Generieren von Zusammenfassungen. Beide dieser Systeme, unterstützen den Benutzer beim Kategorisieren von Dokumenten, ohne deren Inhalt genau kennen zu müssen.
Diese Arbeit stützt sich hauptsächlich auf „Data Mining: Concepts and Techniques“ [3] und „A Survey of Text Summarization Extractive Techniques“ [2].

In dieser Arbeit soll dargelegt werden, wie die Entwicklung eines Text Mining Systems aussieht. Dabei werden zwei Ziele:
1) Die Entwicklung eines Text-Klassifizierungs Modell
2) Das automatische Generieren von Zusammenfassungen.

Auf Basis von Recherchen und Tutorials soll Wissen im Bereich Text Mining

aufgebaut werden und dieses soll gleich zur Entwicklung eines Text Mining Algorithmus verwendet werden. Dabei sollen anhand Experimenten die verschiedenen Prozessschritte evaluiert werden. Unter Berücksichtigung der Genauigkeit der Klassifikation soll der Prozess best-möglich optimiert werden.
Darauf aufbauend wird im zweiten Teil der Arbeit unter Berücksichtigung der ermittelten Schritte ein System beschrieben, welches wissenschaftliche Texte zusammenfassen können soll. Dabei sollen die wichtigsten Sätze eines Dokumentes selektiert werden. Dabei wird nur das Generieren von extraktiven Zusammenfassung behandelt, abstraktive werden in dieser Bachelorarbeit nicht behandelt.
Des Weiteren ist zu berücksichtigen, dass es sich bei dieser Arbeit um eine wissenschaftliche Abhandlung zum Thema Text Mining handelt.

Die Arbeit ist in sechs Kapitel gegliedert. Das erste Kapitel beschäftigt sich mit der Definition der involvierten Fachbereiche. Danach werden die theoretischen Grundlagen erklärt, wie sie in dieser Bachelorarbeit benutzt werden. Im Anschluss werden die verwendeten Tools und die Daten vorgestellt. Im Hauptteil werden dann verschiedene Prozesse beschrieben und deren Resultate dokumentiert. Abschliessend werden nochmal die wichtigsten Punkte dieser Bachelorarbeit aufgezeigt.

Kapitel 2

Definition: Text Mining

Diese Kapitel dient zur Erläuterung des Text Mining.
Text mining ist ein Prozess zur Extraktion von interessantem und nicht-trivialem Wissen aus unstruktierten, bzw. schwach-strukturierten Texten. Dieser Prozess ist eine interdisziplinäre Wissenschaft, so sind folgende Fachbereiche in Betracht zu ziehen: Informations Retrieval (Informationsrückgewinnung), Data mining, Maschinelles Lernen, Statistik und Computerlinguistik.
Die aus dem Text Mining erhaltenen extrahierten Informationen können weiterverwendet werden für Text Analysen wie: Clusteranalysen, Klassifizierung von Texten, Aufbau eines Fragen-Antworten-Systems.

2.1 Data Mining

Data Mining umfasst die Extraktion bzw. das "Mining"von Wissen aus einer grossen Menge von Daten. [3]. Data Mining ist ein Unterbegriff von **KDD**, Knowledge Discovery in Databases, beziehungsweise **KDT**, Knowledge Discovery in Textual Databases und zwar geht es bei Data Mining hauptsächlich darum, sogenannte neue Muster, Data Patterns, zu finden. Diese Patterns werden erst in einem weiteren Schritt evaluiert. Data Mining auf Textbasen soll Schlüsselwörter/Konstrukte extrahieren, welche die Texte so gut wie möglich beschreiben. Für diese Aufgabe

muss Data Mining eine Integration eines Information Retrieval Systems haben, aber auch linguistische Modelle und Klassifikationssysteme berücksichtigen.

Knowledge Discovery ist definiert als eine nicht-triviale Extraktion von impliziten, vorher unbekannten und potentiell nützlichen Informationen von gegebenen Daten. [4]

2.2 Information Retrieval

Information retrieval (IR) ist das Finden von Material (normallerweise Dokumente) unstrukturierter Natur (normallerweise Text) aus einer grossen Kollektion (normallerweise auf Computern gespeichert), welches ein Bedürfnis nach Information stillt.[5]
Anhand dieser Definition ist ersichtlich, dass es bei Informations Retrieval um das Finden von Informationen geht. Dabei soll das Infromationsbedürfnis des Benutzers gestillt werden. Jedoch ist die Dimension von Information Retrieval um einiges grösser als in der Definition angegeben. Nebst unstrukturierten Daten werden auch semi-strukturierte, Bilder und Musik als Inhalt für ein IR-System verwendet. Im Gegensatz zu Data Mining werden die Informationen in Informations Retrieval mit Hilfe von Regeln extrahiert.

2.2.1 Relevanz

Eine wesentliche Herausforderung bei Information Retrieval besteht im Auffinden von relevanten Informationen. Enthält ein Dokument Informationen bezüglich eines frei zu wählenden Kontextes, so nennt man dieses Dokument relevant bezüglich dieses Kontextes.
Die Relevanz kann aber auch von weiteren Faktoren abhängig gemacht werden, beispielsweise vom Veröffentlichungsdatum, vom Stil, von der Sprache oder vom

Urheber. [6]
Um diese Relevanz sichtbar machen zu können, werden sogenannte Retrieval Modelle definiert. Diese Modelle beschreiben meist die statistischen Eigenschaften eines Textes, sie können jedoch auch linguistischer Natur sein.

1. Wörterhäufigkeit, danach Gewichtung mit Hilfe von *tfidf* (Kapitel 3.1.9)
2. Linguistische Features (*POS Tagging, NER,...*)

2.3 Maschinelles Lernen

Das Ziel von Text Mining ist hauptsächlich das maschinelle Bestimmen des Outputs bei gegebenen Daten. Dafür werden wahrschinlickeitstheoretische Modelle angewendet, um eine Prognose zu optimieren. Mit Hilfe von maschinellem Lernen sollen Regel gefunden werden, um Prognosen zu erstellen.

2.4 Computer Linguistik

Unter Computer Linguistik wird eine Verarbeitung natürlicher Sprachen in Computersystemen verstanden. Nennenswert in diesem Zusammenhang wäre das Saarbrücker Pipelinemodell. Dieses beschreibt, wie ein Text maschinell verarbeitet werden soll und welche Möglichkeiten es gibt, um Texte als Vektoren darstellen zu können: Die in Abbildung 2.1 beschriebenen Schritte werden noch etwas genauer

Abbildung 2.1: Pipelinemodell der Sprachverarbeitung[1]

im Kapitel 3.1 Preprocessing beschrieben.

Kapitel 3

Grundlagen

In diesem Kapitel werden die Prozessschritte erläutert, welche in dieser Bachelorarbeit eingesetzt werden. Des Weiteren werden alle relevanten Schlüsselwörter definiert.

3.1 Preprocessing

Bevor irgendwelche Operationen auf Textdaten ausgeführt werden können, muss ein Dokument vorbereitet werden. Diese Datenaufbereitung wird *Preprocessing* genannt. Ein Text in der Grundfassung weist viele irrelevante und störende Elemente auf, welche zuerst entfernt werden müssen. Weiter ist es von Vorteil Wörter auf deren Stamm zu reduzieren, damit sie besser gruppiert werden können und somit die Daten zusätzlich reduziert werden.[7] Ein weiterer Bestandteil von *Preprocessing* ist das Dokument in ein Vektorraum-Modell zu bringen, so dass der Text für mathematische Operationen gebraucht werden kann.

3.1.1 Tokenization

Dieser Schritt ist auch bekannt als Textnormalisierung. Damit ein Dokument für Berechnungen verwendet werden kann, wird er in seine Einzelteile zerlegt. Je nachdem kann es Sinn machen, ein Dokument in Sätze zu unterteilen, meist jedoch ist es notwendig ihn in Wörter zu zerlegen. Ein Dokument d wird nach diesem Prozess wie folgt beschrieben: $d = w_1, w_2, ..., w_n$ und die Vektorkomponenten den einzelnen Wörtern entsprechen. Ihre Nummerierung steht für ihre Position im Text. Die Interpunktion richtig zu interpretieren, ist eine wesentliche Herausforderung. (z.B. Bei Satztokenisierung sollte “etc.“ keine Trennung bewirken.)
Beispiel:
$w_1\ w_2\ w_3\ w_4. \rightarrow (w_1, w_2, w_3, w_4)$

Token: Durch die Tokenisierung wird ein Text in Tokens/Terms unterteilt. Diese Tokens können je nach Fall, ganze Fragmente oder einzelne Wörter sein.

3.1.2 Filterung von Stopwords

Als Stopwörter werden Wörter bezeichnet, welche auch in anderen Texten am häufigsten vorkommen. Diese Wörter können in den meisten Fällen eliminiert werden, da sie oft keine Information beinhalten. Der Vorteil dieses Schrittes ist eine Reduktion der Datenmenge.
Ein Löschen dieser Wörter kann auf verschiedene Arten erfolgen:

1. Bei der Bearbeitung von mehreren Dokumenten können die häufigst vorkommenden Wörter gelöscht werden.
2. Löschen eines Wortes, falls es in einer Stopword-Dictionary vorkommt.

Das ungeprüfte Löschen von Stopwörtern kann aber auch problematisch sein (beispielsweise kann ”nicht”bei Opinion Mining von Bedeutung sein.)

Beispiele für solche Stopwörter sind: *'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if'* (Auszug aus nltk corpus stopwords in Englisch).

3.1.3 Gross/Kleinschreibung Ignorieren

Die Tokens, welche bis auf die gross- und kleinschreibung identisch sind, werden zusammengefasst. Einfachheitshalber wird dieser Schritt nachfolgend als Transform Cases bezeichnet. Die Buchstaben werden alle in kleine Buchstaben transformiert. *Cases* $\rightarrow$ *cases*

3.1.4 Token Filterung (Wortlänge)

Zur weiteren Reduktion der Tokens werden Filter angewendet, welche zu lange oder zu kurze Tokens löschen. Dabei werden hauptsächlich Abkürzungen gelöscht. Gemäss Kumar [7] wäre es sinnvoller, diese Akronyme in ihre Standartform zu bringen. Jedoch ist ein solcher in dieser Bachelorarbeit nicht möglich. Des Weiteren ist zu beachten, dass es je nach Sprache möglich ist, Wörter aneinander zu reihen und so beliebig lange Wörter zu generieren. Dieser Fall kann jedoch in dieser Bachelorarbeit ausgeschlossen werden, da die Dokumente in englischer Sprache sind und zu lange Wörter eher selten vorkommen, somit irrelevant sind.

3.1.5 POS Tagging

Ein *Part-Of-Speech-Tagger* (POS) liest ein Text und fügt zu jedem Wort ein Part-Of-Speech Etikett hinzu, wie Substantiv, Verb, Adjektiv[8]. Der Tagger ermöglicht somit eine Selektion der Wortarten, die für den weiteren Prozess berücksichtigt werden sollen. Dies ist insofern sinnvoll, da Adverben, Interjektionen und andere Wortarten für die Text Klassifikation keine Bedeutung haben.

3.1.6 Stemming

Wörter können in verschiedenen Formen vorkommen, diese sollten best möglich zusammengefasst werden, so dass die Dimension reduziert werden kann. In erster Linie ist es notwendig, Verben auf ihre Infinitiv-Form zurück zu bringen. Zusätzlich sollen Pluralformen singular gemacht werden. In dieser Arbeit wird hauptsächlich der Algorithmus von Porter[9] angewendet. Porter definierte einige Regeln, welche ein englisches Wort in sein *Wortstamm* transformiert.

running $\Rightarrow$ *run*

entities $\Rightarrow$ *entiti*

3.1.7 WordNet Integration

WordNet[1] ist eine Sprachdatenbank in Englisch. Die Wörter sind gruppiert in Sets von Synoymen (Synsets). Durch die Integration dieser Datenbank ist es möglich ähnliche Begriffe, bzw. Wörter mit demselben Überbegriff zusammen zu fassen. Beispiel für solche Synsets anhand von *dog*

Code in Python:

```
from nltk.corpus import wordnet as wn
syns = wm.synsets('dog')
[s.lemmas[0].name for s in syns]
```

Ausgabe:

['dog', 'frump', 'dog', 'cad', 'frank', 'pawl', 'andiron', 'chase']

3.1.8 N-Grams

Bei Texten kann die Reihenfolge der Wörter eine enorme Rolle spielen. Zum Teil ist es deshalb sinnvoll, nicht nur unigrams zu berücksichtigen sondern auch bigrams und trigrams.

[1] http://wordnet.princeton.edu/

$d = w_1\ w_2\ w_3$.

unigram: (w_1, w_2, w_3)

unigram und bigram: $(w_1, w_1w_2, w_2, w_2w_3, w_3)$

Dies erhöht die Dimension der Repräsentation eines solchen Dokumentes, jedoch kann dabei die Genauigkeit der Vorhersage zunehmen. Da zusammengesetzte Ausdrücke wie *Text Mining* mit berücksichtig werden und nicht nur die Wörter einzeln, *Text* und *Mining*.

3.1.9 Vektorraum-Modell

Dieses Modell wird erstmals von Salton [10] beschrieben und repräsentiert ein Dokument unter Berücksichtung von gewissen Statistiken. Es wird im Abschluss des *Preprocessing* generiert.

Die einzelnen Tokens sollen mit der Häufigkeit ihres Auftretens verbunden werden, so dass jedes einzelne Token gewichtet ist.

Gegeben sei ein Dokument $d_j = (w_1, w_2, w_1...w_n)$, wobei gilt, dass bei gleichem Index das Wort gleich ist. Dieses Dokument gilt es nun in die Form eines gewichteten Vektors v_d zu bringen. Dafür gelten folgende Definitionen:

$$tf(t,d) = \frac{f(t,d)}{max\left\{f(w,d) : w \in d\right\}} \tag{3.1}$$

wobei $f(t,d)$ die Häufigkeit des Terms t in d ist. Mit dem Schritt (3.1) werden die Häufigkeiten normalisiert. So können Verzerrungen der Ergebnisse verhindert werden. Der Wertebereich der Gewichtung eines Tokens liegt somit zwischen 0 (Token kommt nie vor) und 1 (Token kommt am häufigsten vor).

Werden wie bei der Klassifikation mehrere Dokumente gleichzeitig betrachtet, soll die Wichtigkeit eines Terms angepasst werden, indem die Bedeutung des Terms für die Gesamtmenge der Dokumente betrachtet wird. Dieser Wert wird inverse Dokumenthäufigkeit *idf* genannt.

$$idf(t,D) = \log \frac{|D|}{1 + |\left\{d \in D : t \in d\right\}|} \tag{3.2}$$

wobei $|D|$ die Anzahl der Dokumente darstellt und $|\{d \in D : t \in d\}|$ die Anzahl der Dokumente d in D ist, bei welchen der Term t vorkommt, also $tf(t,d) \neq 0$.
Die Gewichtung einer Termes unter Berücksichtigung der inversen Dokumenthäufigkeit (3.2) sieht dann folgendermassen aus (3.3).

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D) \tag{3.3}$$

Beispiel: $d = (w_1, w_2, w_1, w_3)$
Nach Gewichtung mittels $tfidf$ (3.3):
$d = \{(w_1, tfidf_{w_1}), (w_2, tfidf_{w_2}), (w_3, tfidf_{w_3})\}$, dabei wird d durch ein Wörterbuch repräsentiert. Ein Eintrag besteht aus dem Wort w als Schlüssel und $tfidf_w$ als Wert und somit der Gewichtung.

Feature/Attribut: Ein Feature bzw. Attribut ist ein solcher Wörterbucheintrag innerhalb des Vektorraum-Modells.

3.2 Postprocessing

Der Nachbearbeitungsprozess in dieser Bachelorarbeit kümmert sich hauptsächlich um die Reduktion des Vektorraum Modells, da ansonsten bei grosser Datenmenge auch die Berechnungszeiten für das erlernen von Vorhersagmodellen zu gross sind. Für die Bestimmung welche Features für die Vorhersage als wichtig erachtet werden, gibt es unterschiedliche Ansätze:

1. Gewichtung nach Korrelation: Die Korrelation beschreibt die Beziehung zwischen Merkmalen.

2. χ^2 Statistik: χ^2 beschreibt die mangelnde Unabhängigkeit zwischen Features und Klassen.

3. Gewichtung nach Gini Index: Statistisches Mass zur Darstellung von Ungleichverteilung.

In dieser Bachelorarbeit wird die Gewichtung nach χ^2 verwendet.

3.2.1 χ^2 Statistik

Gemäss Aggarwal [11] ist χ^2-Statistik ein Weg um die mangelnde Unabhängigkeit zwischen einem Term t und einer Klasse c zu berechnen, ähnlich wie Mutual Information. Jedoch hat die Verwendung von χ^2 den Vorteil, dass die Werte normalisiert werden.

$$\chi_c^2(t) = \frac{n * F(t)^2 * (p_c(t) - P_c)^2}{F(t) * (1 - F(t)) * P_c * (1 - P_c))} \tag{3.4}$$

dabei gilt: Die Trainingsdaten sind bereits an eine Klasse gekoppelt, insofern kann

n	Anzahl der Dokumente
c	Beliebige Klasse aus $C = \{c_1, c_2, ...c_j\}$, $c \in C$
$p_c(t)$	Bedingte Wahrscheinlichkeit von c für Dokumente welche t enthalten, $P(t\|c)$
P_c	Globaler Bruch der Dokumente, welche die Klasse c enthalten
$F(t)$	Globaler Bruch der Dokumente, welche den Term t enthalten.

mit χ^2 Statistik berechnet werden, wie die einzelnen Attribute mit der Klasse korrelieren.

3.2.2 Attribut Reduzierung

Mit Hilfe der berechneten Korrelationen zwischen Klassen und den einzelnen Attributen ist es nun möglich die Dimension eines Dokumentenvektors zu senken. Durch eine Wahl eines optimalen Schwellwertes, können die Attibute, welche zur Vorhersage benötigt werden, selektiert werden und die anderen gelöscht werden. Je nach Datensatz kann dieser optimale Schwellwert variieren.

3.3 Clustering Algorithmen

Der Prozess, ein Set von physikalischen oder abstrakten Objekten in Klassen von ähnlichen Objekten zu gruppieren, wird Clustering genannt. [7]
Bei Clustering sind die Klassen nicht bekannt, jedoch ist bei den meisten Implementationen eine Vordefinition der Clusterzahl notwendig.

3.3.1 K-Means

K-Means ist ein Verfahren, welches für Clustering verwendet werden kann. K-Means konvergiert schnell. In dieser Bachelorarbeit wird folgende Implementierung des K-Means-Algorithmus verwendet:

```
input: Graph g, Seeds seeds, Documents docs
finished ← False
cluster ← []

while (!finished):
        cluster ← []
        finished ← True

        for d in docs:
                cluster.append(generateClustersAroundSeeds(docs, seeds, g))
        for s in seeds:
                newCenters ← calculateNewCenters(cluster)

                if newCenters ≠ seeds:
                        seeds ← newCenters
                        finished ← False

return cluster
```

Die Funktion generateClustersAroundSeeds weist die Dokumente mit Hilfe der Cosine-Similarity (3.5.2) dem nächsten Zentrum zu. Die Funktion calculateNewCenters berrechnet die neuen Zentren auf Grund der Dokumente, welche dem Cluster zugewiesen worden sind.

3.4 Klassifikation

Die Klassifikation ist verwandt zum Clustering. Jedoch sind bei der Klassifikation die Klassen bereits definiert. Die Aufgabe besteht darin, Dokumente einer oder mehreren Klassen zuordnen. Mit Trainingsdaten soll gelernt werden, welche Features zu welchen Klassen tendieren. In dieser Bachelorarbeit werden verschiedene Klassifikationsalgorithmen miteinander verglichen. Diese Klassifikationsalgorithmen werden in dieser Arbeit nicht selbst implementiert.

Classifier: Ein Classifier ist ein Entscheider, den es zu lernen gilt. Dieser Entscheider soll nach dem Lernverfahren in der Lage sein Daten zu klassifizieren.
Overfitting: Overfitting ist ein zu starkes optimieren des Classifiers durch die Trainigsdaten. Der Classifier wird für die Vorhersage der Trainingsdaten zwar besser, für das Bestimmen von anderen Daten wird er jedoch schlechter.

3.4.1 Support Vector Machine

Die Support Vector Machine (SVM) unterteilt eine Menge von Objekten durch die Erstellung einer Hyperebene in Klassen. [12] Eine Hyperebene ist definiert durch den Normalenvektor w und die Verschiebung b:

$$HyperebeneH = \{x|\langle w, x\rangle + b = 0\} \tag{3.5}$$

Das Training erfolgt nun mit der Bestimmung von w und b, so dass die Hyperebene H die Trainingsdaten trennt. Zur Vorhersage muss nur herausgefunden werden, auf welcher Seite sich der neue Punkt befindet.
Bei der SVM wird versucht die Hyperplane zu setzen, so dass die Trennspanne maximal ist. Fehler werden zugelassen, aber bestraft.

3.4.2 Naïve Bayes

Der Naïve Bayes Classifier geht von einer Unabhängigkeit zwischen Features aus [13]. Will heissen, dieser Classifier geht davon aus, dass die Anwesenheit (oder Abwesenheit) eines bestimmten Features einer Klasse ist ohne Bezug zur Anwesenheit (oder Abwesenheit) eines jeglichen anderen Features. [14]
Der Classifier basiert auf der Bayes'schen Regel:

$$P(h|D) = \frac{P(D|h) * P(h)}{P(D)} \tag{3.6}$$

hierbei ist: $P(h)$ Wahrscheinlichkeit, dass h aus H gültig ist (a priori). $P(D)$ Wahrscheinlichkeit, dass D als Ereignisdatensatz auftritt (ohne die gültige Hypothese zu kennen). $P(D|h)$ Wahrscheinlichkeit des Auftretens von D unter der Bedingung, dass h eingetreten ist. $P(h|D)$ Wahrscheinlichkeit, dass h wahr ist bei gegebenen beobachteten Daten D (a posteriori).

Die Aufgabe des Classifiers besteht nun darin, anhand einer gegebenen Menge an klassifizierten Daten die Wahrscheinlichkeiten zu berechnen und dann die wahrscheinlichste Klasse c_{max} für ein neues Dokument D zu ermitteln.

$$c_{max} = max_{c \in C} P(c|D) \tag{3.7}$$

dabei gilt, C ist eine Endliche Menge von Klassen $C = \{c_1, ..., c_n\}$ und D ist eine Konjunktion von Attributen/Features $D = \langle a_1, ..., a_m \rangle$. [15]

3.4.3 ID3/Decision Tree

ID3 ist eine andere Form eines Decision Trees (Entscheidungsbaum). Der Entscheidungsbaum soll anhand von gegenbenen Daten mit gleichen Attributen entscheiden zu welcher Klasse ein Tupel gehört.
Ein Knoten im Entscheidungsbaum steht für ein nicht-kategorisierendes Attribut und jeder Ast einem möglichen Wert des Attributes. Die Blätter des Baumes sind

die verschiedenen Klassen und ein Tupel der Daten wird beschrieben durch den Weg von der Wurzel des Baumes zu einem Blatt.
An jedem Knoten wird der Ast genommen, welcher am *meisten Infofmationen* hat. [16]

3.5 Evaluation

Für die Findung des Optimums ist es notwendig, die Resultate, die Modelle und die Vorhersagen zu bewerten. Für unterschiedliche Aufgaben sind unterschiedliche Messgrössen vorhanden.
Die Relevanzbewertung wird hauptsächlich für Information Retrieval verwendet, sie kann auch die Prognose eines Klassifikationsmodells bewerten.
Die Similarity Bewertung wird in dieser Bachelorarbeit hauptsächlich für die Bewertung der generierten Zusammenfassungen verwendet.

3.5.1 Relevanzbewertung

Evaluiert wird mit Hilfe der beiden Scoring-faktoren, *Recall* und *Precision*. Damit die Begriffe *precision* und *recall* verstanden werden können, ist eine Definition von *true positive*, *false positive*, *true negative* und *false negative* erforderlich:
Gegeben sind zwei Klassen (c_1 und c_2), dabei wird c_1 als die positive und c_2 als die negative Klasse betrachtet. Nach der Vorhersage sind folgende Werttypen zu untersuchen:

true positive	Anzahl Dokumente, die zur Klasse positive gehören und auch ihr zugewiesen worden sind.
false positive	Anzahl Dokumente, die zur Klasse positive gehören, aber vom Classifier der Klasse negative zugewiesen worden sind.
false negative	Anzahl Dokumente, die zur Klasse negative gehören, aber vom Classifier der Klasse positive zugewiesen worden sind.
true negative	Anzahl Dokumente, die zur Klasse negative gehören und auch ihr zugewiesen worden sind.

Tabelle 3.1: Definition von tp, fp, fn und tn.

Die Scores *precision* und *recall* werden wie folgt definiert:

$$precision = \frac{tp}{tp + fp} \tag{3.8}$$

$$recall = \frac{tp}{tp + fn} \tag{3.9}$$

wobei gilt: tp = true positive, fp = false positive, fn = false negative.

Die Interpretation dieser Scores lautet wie folgt aus:
precision: Wahrscheinlichkeit, dass ein zufällig selektiertes, vorhergesagtes Dokument auch relevant ist.
recall: Wahrscheinlichkeit, dass ein zufällig selektiertes, relevantes Dokument auch im Set der Vorhergesagten ist.

Eine weitere Möglichkeit die Klassifikation zu bewerten, ist die *accuracy*.

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{3.10}$$

wobei tn = true negative. Der Wert der *accuracy* liegt zwischem 0 (schlechte Klassifikation) und 1 (optimale Klassifikation).

Das Mass *accuracy* weist allerdings gewisse Schwierigkeiten auf. Sind die Dokumente nur zu einem Bruchteil relevant, so kann der Wert doch relativ hoch sein, obwohl der Classifier ungenügend ist. In dieser Bachelorarbeit ist die *accuracy* als ausreichend betrachtet.

3.5.2 Similarity Bewertung

Bewertet werden die generierten Zusammenfassungen mit Hilfe eines *Similarity* Vergleiches mit einem gegebenen Abstract. Als *Similarity-Score* wird Cosine-Similarity (3.11) verwendet.

$$cos_{dist}(A, B) = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}} \tag{3.11}$$

wobei A und B je Termfrequenz-Vektoren von zwei verschiedenen Dokumenten sind. Geht cos_{dist} gegen 1 so sind die beiden Dokumente nahezu identisch und je mehr dieser Wert gegen 0 strebt, desto weniger haben die beiden Vektoren und somit ihre Dokumente gemein.

3.5.3 Cross-Validation

Bei der *Cross-Validation* werden die Trainingsdaten in Segmente unterteilt. Im Lernprozess des Classifiers wird je Iteration ein zufälliges Segment zur Evaluation und die übrigen Segmente für sein Training verwendet. Dieser Schritt wird beliebig oft angewandt.

Kapitel 4

Tools

In diesem Kapitel werden die für die Bachelorarbeit verwendeten Tools vorgestellt. Dabei werden die Stärken wie auch deren Schwächen ersichtlich.

4.1 Rapidminer

Rapidminer ist eine Open-Source Software, welche Aufgaben der Wissensentdeckung unterstützt. Dabei werden folgende Schritte ermöglicht:

1. Ein- und Ausgabe
2. Preprocessing
3. Maschinelles Lernen
4. Data Mining
5. Text Mining
6. Web Mining
7. ...

Rapidminer wurde in Java geschrieben und ist somit auf den meisten Betriebssystemen verwendbar. Rapidminer ist mit Java auch erweiterbar, so können problemlos weitere Algorithmen geschrieben werden.
Der User kann mit Hilfe von Operatoren relativ einfach Prozesse nachbilden. Mit der Verkettung von solchen Operatoren zu sogenannten Operatorbäumen ist ein Datenfluss garantiert (Abbildung 4.1).
Mittels Drag-and-Drop können diese Bäume modelliert werden, jedoch ist es auch möglich sie mit XML zu definieren.
Rapidminer erlaubt es auch Funktionen von R und WEKA zu benutzen.

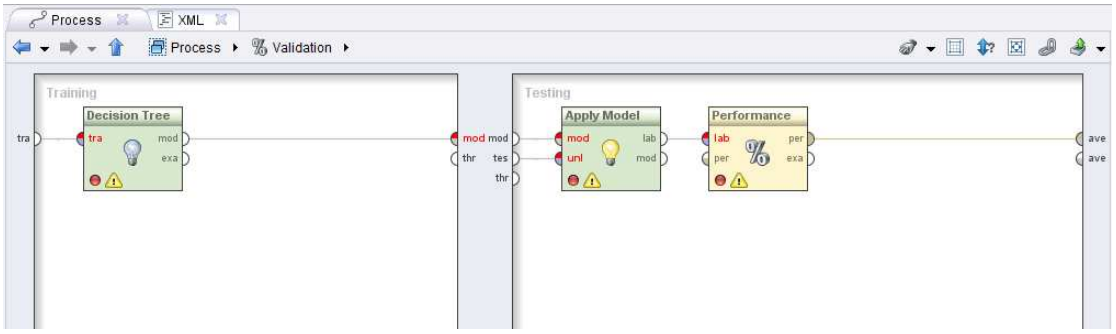


Abbildung 4.1: Beispiel von Operatoren in Rapidminer.

4.2 Python

Für den zweiten Teil der Bachelorarbeit wird vorwiegend Python benutzt, damit sämtliche Prozesse selbst definiert werden können. Dadurch sind die Schritte besser überschaubar und können auch besser angepasst werden. Bei Python handelt es sich um eine dynamische Programmiersprache und wird im Rahmen dieser Bachelorarbeit als Scriptsprache verwendet.

4.3 NLTK - Natural Language Toolkit

Damit Operationen aud der Natürlichen Sprach Bearbeitung benutzt werden können, empfiehlt es sich Natural Language Toolkit zu verwenden. Dieses Toolkit bietet

nebst Text Corpi auch noch Tokenizer, POS Tagging und andere Sprachanalyse Werkzeuge.

Kapitel 5

Datensatz

In diesem Kapitel werden die für die Bachelorarbeit verwendeten Datensätze vorgestellt.

5.1 Classification

Bei der Classification liegen zwei Datensätze zugrunde. Beide Datensätze weisen die gleichen Labels (Klassenzurodnungen) auf, unterscheiden sich jedoch in der Anzahl Dokumente pro Labels und in den Dokumenten selbst.
Bei den Dokumenten handelt es sich um Newsgroups. Ein Dokument besitzt einen Headerteil, welcher für die Verarbeitung nicht berücksichtigt werden soll und einem Inhalt. Der Inhalt kann sehr kurz sein, was die Klassifizierung erschweren kann. Ein Beispieltext ist in Abbildung 5.1 zu sehen.

Datensatz	Klassenzahl	Dokumente pro Klasse
mini_newsgroups	20	100
20_newsgroups	20	1000

Tabelle 5.1: Datensätze

```
Path: cantaloupe.srv.cs.cmu.edu!crabapple.srv.cs.cmu.edu!fs7.ece.cmu.edu!eur
From: weston@ucssun1.sdsu.edu (weston t)
Newsgroups: comp.graphics
Subject: graphical representation of vector-valued functions
Date: 5 Apr 1993 20:22:28 GMT
Organization: SDSU Computing Services
Lines: 13
Message-ID: <1pq4e4$afc@pandora.sdsu.edu>
NNTP-Posting-Host: ucssun1.sdsu.edu

gnuplot, etc. make it easy to plot real valued functions of 2 variables
but I want to plot functions whose values are 2-vectors. I have been
doing this by plotting arrays of arrows (complete with arrowheads) but
before going further, I thought I would ask whether someone has already
done the work. Any pointers??

thanx in advance


Tom Weston                    | USENET: weston@ucssun1.sdsu.edu
Department of Philosophy      | (619) 594-6218 (office)
San Diego State Univ.         | (619) 575-7477 (home)
San Diego, CA 92182-0303      |
```

Abbildung 5.1: Beispiel eines Dokumentes aus der Klasse comp.graphics.

5.2 Automische Zusammenfassung

Für das Testen des definierten Algorithmus wird die Wissenschaftliche Arbeit von Vishal Gupta und Gurpreet Singh Lehal: *A Survey of Text Summarization Extracitve Techniques* verwendet. [2]

Kapitel 6

Umsetzung und Resultate

In diesem Kapitel werden verschiedene Testreihen dokumentiert und die erzielten Resultate werden erklärt.

6.1 Klassifikation I

6.1.1 Preprocessing I

Die Daten sind in dieser Testreihe immer dieselben, es sind hier um zwei Klassen aus dem Datensatz "min_newsgroups":

	sci.electronics	sci.med
Anzahl der Dokumente	100	100

Tabelle 6.1: Klassifikation I, Preprocessing I, Datensatz

Diese Dokumente werden als Trainings- und Testdaten zugleich verwendet. Will heissen, der Klassifizierer wird mittels Cross-Validation bewertet.

6.1.1.1 Prozess I

In einem ersten Schritt wird ein Tokenization angewendet. Dabei wird ein Dokument unterteilt in Fragmente, sobald ein Zeichen auftaucht, welches keinen Buchstaben repräsentiert (Tokenizing by non-letters).

Preprocessing	Tokenization, TFIDF
Anzahl Features	12'079
Klassifikation	Naïve Bayes
Resultat	Es wird eine Accuracy von 93% erreicht.
Beobachtung	Die Accuracy ist schon relativ gut, für dass das es sich nur um kleine Dokumente handelt, und nur mit 200 Dokumenten trainiert wird. Problematisch könnte die Menge der Features werden, da der Klassifikationsalgorithmus durch eine hohe Featurezahl verlangsamt wird. Alle Features sind noch in der Form, wie sie im Dokument erscheinen.

Tabelle 6.2: Klassifikation I, Prozess I

6.1.1.2 Prozess II

Zusätzlich wird ein Filter angewendet. Dieser Filter entfernt alle Fragmente, welche eine Minimallänge von 4 unterschreiten bzw. eine Maximallänge von 25 überschreiten.

Preprocessing	Tokenization, Filter Tokens nach Länge [4; 25], TFIDF
Anzahl Features	10'623
Klassifikation	Naïve Bayes
Resultat	Es wird eine Accuracy von 92.5% erreicht.
Beobachtung	Die Genauigkeit der Vorhersage wurde in diesem Schritt nicht gravierend reduziert. Diese Methode ist effektiv um die Dimension zu reduzieren

Tabelle 6.3: Klassifikation I, Prozess II

6.1.1.3 Prozess III

Nebst dem Filtern von zu kurzen bzw. zu langen Wörtern wird ein Stopword-Filter eingesetzt.

Preprocessing	Tokenization, Filter Tokens nach Länge [4; 25], Filtern von Stopwords, TFIDF
Anzahl Features	10'219
Klassifikation	Naïve Bayes
Resultat	Es wird eine Accuracy von 92% erreicht.
Beobachtung	Die Genauigkeit hat auch bei diesem Schritt etwas abgenommen.

Tabelle 6.4: Klassifikation I, Prozess III

6.1.1.4 Prozess IV

Da für zwei Klassen immer noch zuviele Features vorhanden sind, wird versucht die Wörtermittels Gleichbehandlung von Gross- und Kleinschreibweise zu gruppieren.

Preprocessing	Tokenization, Filter Tokens nach Länge [4; 25], Filtern von Stopwords, Transform Cases, TFIDF
Anzahl Features	8'825
Klassifikation	Naïve Bayes
Resultat	Es wird eine Accuracy von 92% erreicht.
Beobachtung	Bei gleichbleibender Exaktheit konnte die Dimension etwas reduziert werden, es erfolgte eine Gruppierung der Features.

Tabelle 6.5: Klassifikation I, Prozess IV

6.1.1.5 Prozess V

Zu den vorherigen Erweiterungen des Preprocessing-Prozesses kommt ein Gruppieren der Features mit Hilfe des Porter Stemmers hinzu.

6.1.1.6 Fazit

Anhand dieser Testreihe, kann aufgezeigt werden, wie sich die einzelnen Vorbereitungsschritte, auf die Features, aber auch auf die Vorhersage/Klassifikation auswirken. Obwohl gewisse Methoden die Vorhersage verschlechtern, sind dennoch alle wichtige Bestandteile des Preprocessings. Beispielsweise wird die Exaktheit

Preprocessing	Tokenization, Filter Tokens nach Länge [4; 25], Transform Cases, Porter Stemming, TFIDF
Anzahl Features	6'543
Klassifikation	Naïve Bayes
Resultat	Es wird eine Accuracy von 94.5% erreicht.
Beobachtung	Durch das Stemming gibt es einen Anstieg in der Exaktheit. Dies ist darauf zurückzuführen, dass gewisse Wörter in zu vielen verschiedenen Formen existieren und ein Gruppieren deren Relevanz gegenüber anderen Tokens gesteigert wird. Zusätzlich konnte die Dimension weiter reduziert werden.

Tabelle 6.6: Klassifikation I, Prozess V

des Vorhersagens durch das Entfernen von Stopwörter verschlechtert Trotzdem macht es Sinn, diese bei dieser Problemstellung zu entfernen, da eine Korrelation zwischen Stopwörter und einzelnen Klassen bei diesem Datensatz eher Zufall ist. Da es sich bei unserem System, nicht um ein Opinion Mining System handelt, dürfte ein Stopwort für die Klassifikation ziemlich irrelevant sein.

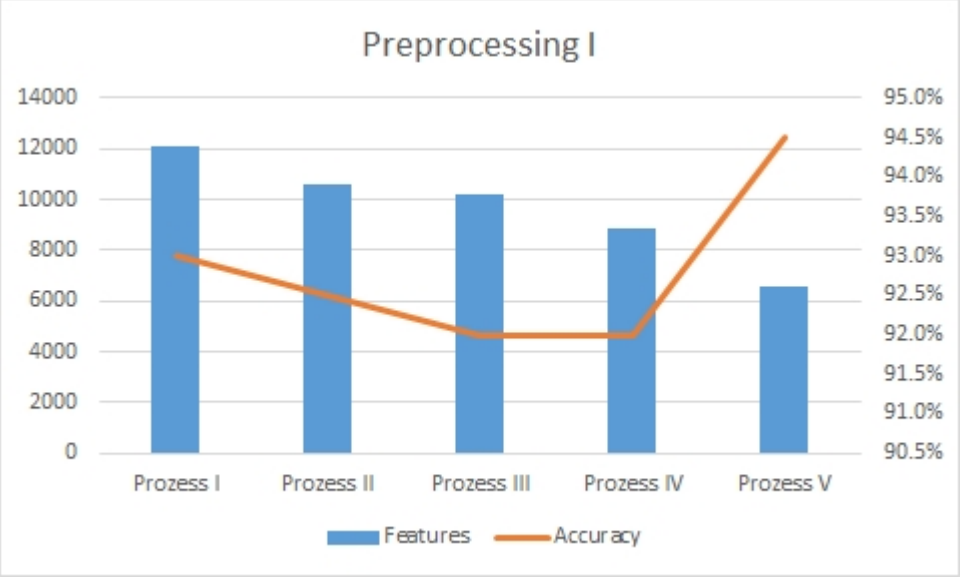


Abbildung 6.1: Preprocessing I: Veränderung der Vorhersage durch Hinzufügen von Teilschritten

6.1.2 Preprocessing II

Diese zweite Testreihe behandelt die Klassifizierung von drei Klassen aus mini_newsgroups: sci.medicine, sci.electronics und sci.space. Der Fokus in dieser Testreihe liegt auf linguistischen Preprocessing Methoden. Die Featuremenge wird nicht beachtet. Da davon ausgegangen wird, dass die Featuremenge genügend minimeirt wurde. Die Evaluation des Klassifiezierungsmodells erfolgt wiederum über die Cross-Validation.

6.1.2.1 Prozess VI

Dieser Prozess beinhaltet die gleichen Schritte wie der Prozess V (6.1.1.5), jedoch wird wie erwähnt mit anderen Dokumenten gelernt und getestet. Dieser Prozess dient zu Vergleichszwecken für die Nachfolgenden.

Preprocessing	Tokenization, Filter Tokens nach Länge [4; 25], Transform Cases, Porter Stemming, TFIDF
Klassifikation	Naïve Bayes
Resultat	Es wird eine Accuracy von 93.67% erreicht.
Beobachtung	Wie zu erwarten war, sank die Genauigkeit der Vorhersage. Zurückzuführen ist dies auf die Tatsache, dass die Bestimmung von drei Klassen komplexer ist als die von zwei.

Tabelle 6.7: Klassifikation I, Prozess VI

6.1.2.2 Prozess VII

Bei diesem Prozess wird erstmals WordNET integriert, indem die Tokens anhand ihres Synsets gruppiert werden.

6.1.2.3 Prozess VIII

Da gewisse Wortarten (wie Interjektionen) für die Klassifikation keine Relevanz haben sollten, wird versucht diese bei der Klassifikation nicht zu beachten. Für

Preprocessing	Tokenization, Filter Tokens nach Länge [4; 25], Transform Cases, Porter Stemming, Synonyms, TFIDF
Klassifikation	Naïve Bayes
Resultat	Es wird eine Accuracy von 94.67% erreicht.
Beobachtung	Durch das erneute Gruppieren von Tokens konnte die Exaktheit weiter gesteigert werden.

Tabelle 6.8: Klassifikation I, Prozess VII

diesen Schritt wird ein POS-Tagger eingesetzt der nur Adjektive, Nomen und Verben erlaubt.

Preprocessing	Tokenization, Filter Tokens nach Länge [4; 25], Transform Cases, Porter Stemming, Synonyms, POS Tagger, TFIDF
Klassifikation	Naïve Bayes
Resultat	Es wird eine Accuracy von 95% erreicht.
Beobachtung	Wie im Vorfeld erwartet, konnte die Genauigkeit der Vorhersage verbessert werden.

Tabelle 6.9: Klassifikation I, Prozess VIII

6.1.2.4 Prozess IX

Zusätzlich zu den Synonymen wird eine Gruppierung nach Unterbegriffen (Hyponyms) hinzugeschaltet.

Preprocessing	Tokenization, Filter Tokens nach Länge [4; 25], Transform Cases, Porter Stemming, Synonyms, Hyponyms, POS Tagger, TFIDF
Klassifikation	Naïve Bayes
Resultat	Es wird eine Accuracy von 95.33% erreicht.
Beobachtung	Auch bei der Anwendung dieser linguistischen Methode konnte die Exaktheit verbessert werden.

Tabelle 6.10: Klassifikation I, Prozess IX

6.1.2.5 Fazit

Anhand der erhaltenen Resultate kann gezeigt werden, dass sprachspezifische Ansätze bei der Klassifikation von Texten nicht vernachlässigt werden soll. Bemerkt wurde

hier aber auch, dass diese Methoden den Prozessablauf deutlich verlangsamen, da bei diesen Methoden stets auf ein Wörterbuch zugegriffen werden muss.

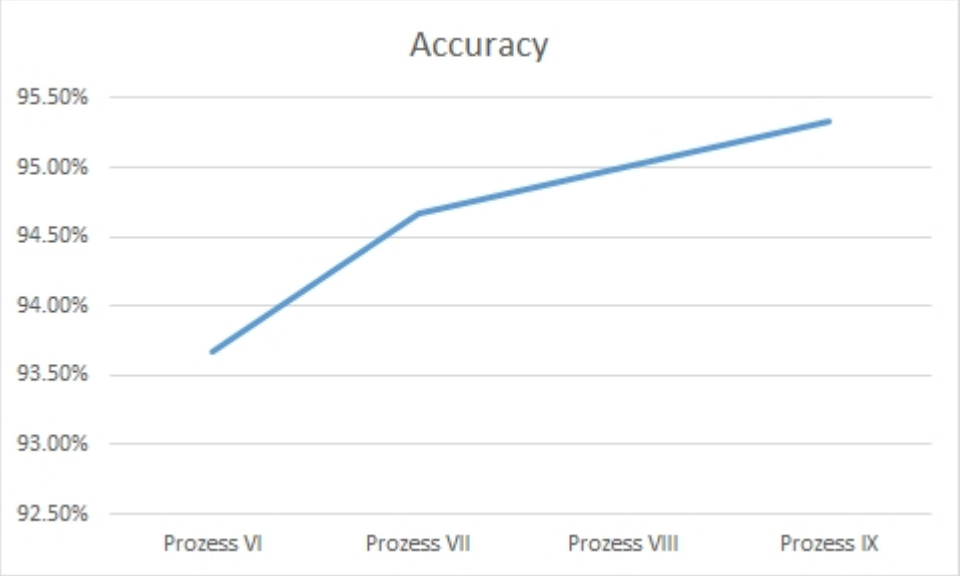


Abbildung 6.2: Preprocessing II: Vergleich von linguistischen Ansätzen

6.1.3 Vergleich von Klassifikationsalgorithmen

Bei dieser Testreihe werden Klassifikationsalgorithmen mit einander verglichen. Für die Datenaufbereitung wird der im Prozess IX (6.1.2.4) beschriebene Ablauf verwendet. Folgende Algorithmen sollen untersucht werden:

1. Naive Bayes
2. Decision Tree
3. ID3
4. SVM

6.1.3.1 Fazit

An denerhaltenen Resultate ist erkennbar, dass die Baumstrukturen die besseren Klassifizierer sind. Jedoch befinden sich alle getesteten Algorithmen auf dem etwa

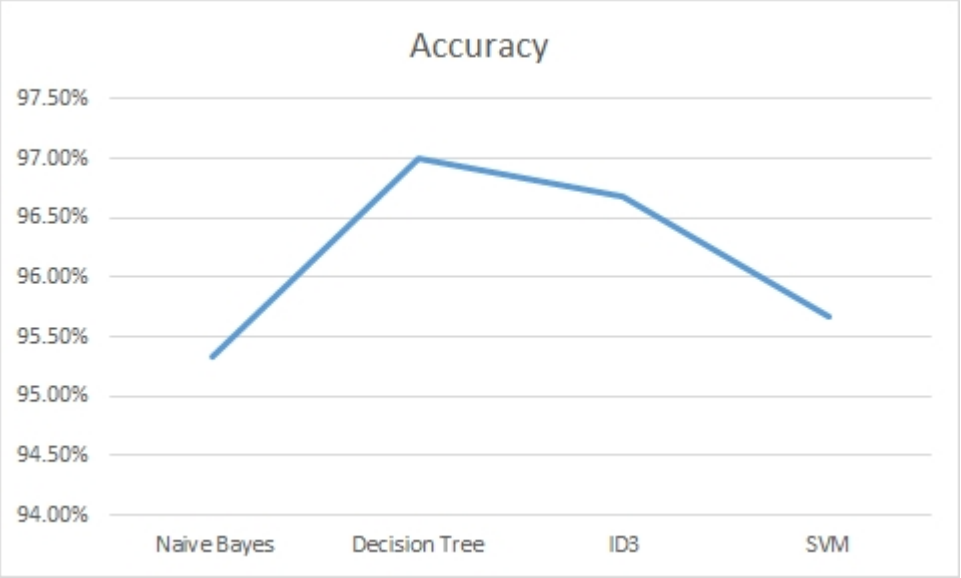


Abbildung 6.3: Klassifikation: Optimierungsversuche

selben Level. Insofern spielt die Wahl des Classifiers für den verwendeten Datensatz keine grosse Rolle.

6.1.4 Weitere Tests

Zudem wurden folgende Einstellungen getestet:

Accuracy	Beschreibung
91.50%	Voraussage von 6 Klassen mit SVM, Preprocessing nach Prozess IX
97.97%	Voraussage von 3 Klassen aus dem 20_newsgroups Datenset mit SVM, Preprocessing aus Prozess IX

Anhand diesen Resultaten ist ersichtlich, dass die Klassifikation der grossen Datenmenge besser ist, als die der kleinen. Die Exaktheit der grösseren Datenmenge hat zwei Ursachen, erstens kann der Classifier mit mehreren Dokumenten (1000 Dokumente pro Klasse) ermittelt werden und zweitens, sind die Dokumente in diesem Datensatz tendenziell länger.

6.1.5 Klassifikation: Grosse Datenmenge bestimmt Kleine

Auf Grund der guten Werte in den vorhergehenden Tests wird ein neuer Versuch gestartet. Bei diesem Versuch soll der Classifier mit drei Klassen aus der Datenmenge 20_newsgroups das Klassifizierungsmodell lernen. Der Classifier wird mittels Crossvalidation optimiert. Mit dem gelernten Modell sollen dann die Dokumente aus der Datenmenge mini_newsgroups bestimmt werden (die Klassen sind dieselben). Für diesen Versuch wird das Preprocessing aus dem Prozess IX (6.1.2.4) verwendet.

6.1.5.1 Resultat

Erreichte Genauigkeit: 33.4%.
Dieser Versuch hat zwei Probleme. Zum Einen waren die Berechnungszeiten für das Bestimmen des Classifiers relativ hoch, will heissen es muss mit zu vielen Features gelernt werden und zum Anderen erreichte die Vorhersage eine schlechte Genauigkeit. Dies ist auf ein Over fitting zurückzuführen.

6.2 Klassifikation II

Da eine Accuracy von 33.4% als ungenügend empfunden wird, wird eine neue Testreihe gestartet, bei welcher das Vorhersagen von Testdaten mit Hilfe der Trainingsdaten im Vordergrund steht. Damit das Problem von Overfitting nicht wieder auftritt, wird keine Crossvalidierung mehr verwendet.

6.2.1 Einstellungen

Für die Vorhersage werden nur zwei Klassen verwendet. Folgende Einstellungen sind gemacht worden:

6.2.1.1 Daten

Trainingsdaten und Testdaten:

Trainingsdaten	Testdaten
Aus 20_newsgroups: sci.med und sci.space	Aus mini_newsgroups: sci.med und sci.space

Tabelle 6.11: Klassifikation II, Daten

6.2.1.2 Preprocessing

Im ersten Schritt wird der Header des Dokumentes entfernt. Danach sieht der Prozess wie folgt aus:
Tokenization → Transform Cases → Filter Stopwords → Generierung von Bigrams.

6.2.1.3 Postprocessing

Da die Datenmenge zu gross ist und der Prozess zu lange dauert, wird erstmals Postprocessing verwendet. Gewichtet wird mit Chi-Square Statistik. Ab welchem Korrelationswert ein Feature für das Lernen verwendet wird, ist unterschiedlich (Schwellwert für χ^2).

6.2.2 Resultate

Prozess	Schwellwert für χ^2	Klassifikations Algorithmus	Accuracy
Prozess X	≥ 1	Naive Bayes	61%
Prozess XI	≥ 0.5	Naive Bayes	80%
Prozess XII	≥ 0.3	Naive Bayes	90%
Prozess XIII	≥ 0.3	SVM	90%
Prozess XIV	≥ 0.3	ID3	89%
Prozess XV	≥ 0.3	Decision Tree	71%

Tabelle 6.12: Klassifikation II, Resultate

In den ersten drei Prozessen X - XII wird versucht die Schwelle zu optimieren. In Prozess XII - XV werden wiederum verschiedene Klassifikationsalgorithmen getestet und verglichen.

6.2.3 Fazit

Durch Postprocessing konnten die Berechnungszeiten enorm verkürzt werden. Schwierig zu erklären ist das schlechte Resultat des Decision Trees in Prozess XV, vorallem da Tree-Classifier bei vorherigen Tests besser waren. Es ist jedoch nicht möglich im Rahmen dieser Bachelorarbeit eine Erklärung für dieses Verhalten zu finden.

6.3 Klassifikation III

Mit den in Klassifikation I und Klassifikation II gesammelten Erfahrungen wird ein finaler Prozess definiert.

Trainingsdaten	Testdaten
20_ newsgroups	mini_newsgroups

Tabelle 6.13: Klassifikation III, Daten

6.3.1 Prozess

Der Prozess ist so definiert,wie in Abbildung 6.4 dargestellt.

6.3.2 Resultate

Wie in der Abbildung 6.4 ersichtlich konnte die Featuremenge von rund 100'000 (bei zwei Klassen) auf rund 400 heruntergebrochen werden. Denoch ist die Vorhersage noch relativ genau:

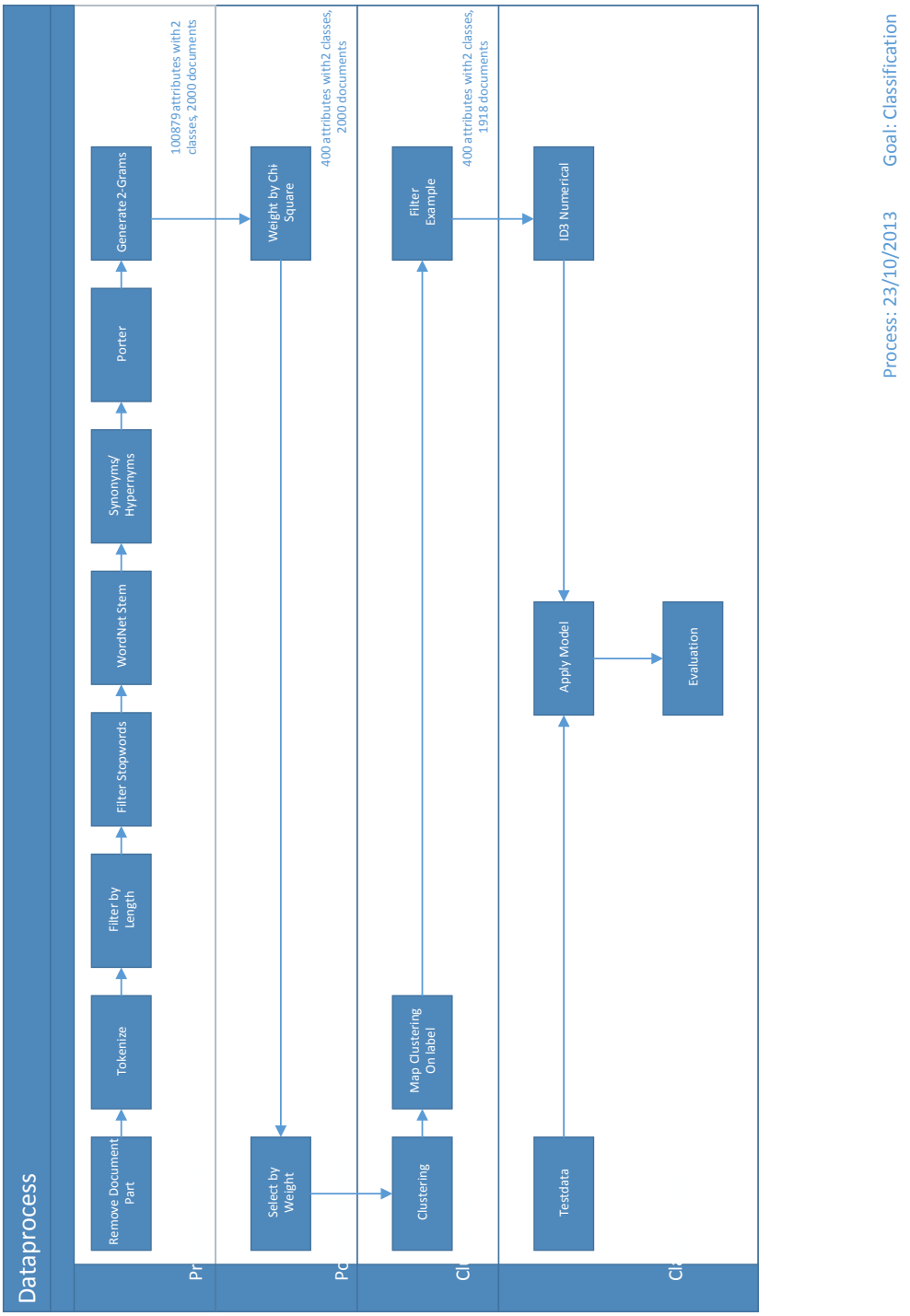


Abbildung 6.4: Finaler Klassifizierungsprozess

	true atheism	true electronics	class precision
pred. atheism	93	8	92.08%
pred. electronics	7	92	92.93%
class recall	93.00%	92.00%	90%

Tabelle 6.14: Klassifikation III: Resultate, 2 Klassen

	true atheism	true space	true electronics	precision
pred. atheism	93	4	8	88.57%
pred. space	6	94	5	89.52%
pred. electronics	1	2	87	96.67%
recall	93%	94%	87%	

Tabelle 6.15: Klassifikation III: Resultate, 3 Klassen

Bei zwei Klassen wird eine Accuracy von 92.5% erreicht.

Bei drei Klassen wird eine Accuracy von 91.3% erreicht.

	true autos	true atheism	true space	true electronics	precision
pred. autos	74	3	0	6	89.16%
pred. atheism	6	91	3	10	82.73%
pred. space	4	2	90	5	89.11%
pred. electronics	16	4	7	79	74.53%
recall	74%	91%	90%	79%	

Tabelle 6.16: Klassifikation III: Resultate, 4 Klassen

Bei vier Klassen wird eine Accuracy von 83.5% erreicht.

6.3.3 Fazit

Die Genauigkeit der Vorhersagen ist in einem guten Bereich, eine weitere Optimierung kann schwierig werden, da die Dokumente relativ kurz sind und somit anhand von wenigen Tokens pro Dokument klassifiziert werden muss.
Da Klassen überlappend sein können oder Dokumente weit enfernt von ihrem Klassenzentrum sein können, ist es sinnvoll, Dokumente, welche sich in diesen Regionen befinden, mit Hilfe von Clustering vom Lernprozess auszuschliessen.

6.4 Automatische Zusammenfassung

Der zweite Teil dieser Bachelorarbeit beschäftigt sich mit dem automatischen Zusammenfassen eines wissenschaftlichen Textes. Für diese Aufgabe wird nicht mehr Rapidminer verwendet, aus Gründen die in Kapitel 6.4.1 genauer erläutert werden.

6.4.1 Systemevaluation

Anhand der Text Klassifikation wurde ersichtlich, dass für komplexere Probleme, wie das automatische Generieren von Zusammenfassungen, Rapidminer nicht genügt. Es sollen Tools eingesetzt werden, die eine flexible Transformation eines Dokumentes erlauben. Jeder Schritt soll korrigiert und angepasst werden können. Des Weiteren kam bei Rapidminer das Problem auf, dass abgespeicherte Zwischenschritte aus unerklärlichen Gründen eine enorme Dateigrösse besassen ($\tilde{1}$5GB) und nichts mehr mit diesen Dateien gemacht werden konnte.
Um diese Probleme zu umgehen, wurde dieser Teil der Bachelorarbeit in Python meist selbst implementiert. Dafür musste Python erlernt werden und die einzelnen Prozessschritte mussten nochmals ausprobiert werden.

6.4.2 Prozess

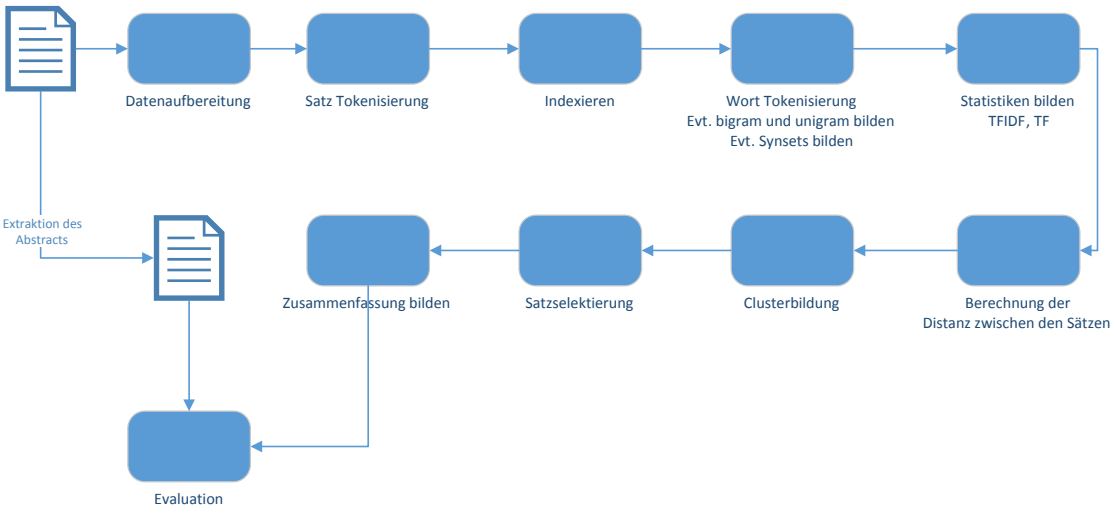


Abbildung 6.5: Automatische Zusammenfassung: Prozess

Beschreibung:

1. **Extraktion des Abstracts:** In diesem Schritt wird das Abstract vom Dokumentenkorpus getrennt. Dieses wird zu einem späteren Zeitpunkt für die Evaluation des Systems benötigt.

2. **Datenaufbereitung:** Das Dokument wird aufbereitet, so dass am Schluss fast nur noch der Text vorhanden ist. Will heissen, die Quellen werden entfernt, die Titel werden ausgelagert, Formeln werden entfernt,... Dieser Schritt ist Dokumentenspezifisch.

3. **Satz Tokenisierung:** Da die Zusammenfassung aus bereits existierenden Sätzen generiert wird, muss der Text in Sätze unterteilt werden.

4. **Indexierung:** Die extrahierten Sätze müssen durchnummeriert werden, damit sie in der richtigen Reihenfolge wieder verwendet werden können.

5. **Wort Tokenisierung:** Die Sätze müssen weiter unterteilt werden, damit ein Gewichten der Sätze erfolgen kann. Bei diesem Schritt soll auch möglich sein, aus Wörtern, Bigramme und/oder Trigramme zu bilden.

6. **Statistik bilden:** Damit die Sätze bewertet werden können, muss auf den Tokens eine Gewichtung stattfinden. Für die Berechnung des IDF-Wertes werden die Sätze als Dokumente angeschaut. Des Weiteren wird eine andere Formel für die TF Berechnung verwendet (6.1).

$$tfs(t, s) = \frac{f(t, d)}{|s|} \tag{6.1}$$

dabei ist t ein Token und s ein Satz. Es wird folglich über die Satzlänge normalisiert, dadurch soll gewährleistet werden, dass lange Sätze nicht automatisch eine höhere Gewichtung haben.

$$idfs(t, S) = \log \frac{|S|}{1 + |\{s \in S : t \in s\}|} \tag{6.2}$$

dabei ist S die Menge aller Sätze in diesem Dokument.

7. **Berechnung der Distanz zwischen Sätzen:** Um ein Clustering zu vereinfachen, werden die einzelnen Sätze paarweise miteinander verglichen. Dabei wird ihre Ähnlichkeit mittels Cosine Similarity berechnet.

8. **Clusterbildung:** Es wird davon ausgegangen, dass ein Dokument mehrere Themengebiete umfasst. Aus diesem Grund sollen Gruppen von Sätzen gebildet werden, die sich ähnlich sind. Eine Gruppe entspricht somit einem Themengebiet. Für das Finden der Gruppenzahl wären tiefere Analysen notwendig.

   Damit ein Dokument gut zusammengefasst werden kann, ist es notwendig, dass sämtliche Themen in der Zusammenfassung vertreten sind.

   Für das Clustering wird in dieser Bachelorarbeit eine Implementation von K-Means verwendet.

9. **Satzselektierung:** Aus den einzelnen Clusters werden Sätze selektiert, welche die höchste Gewichtung haben. Je nach Clustergrösse, kann es sinnvoll sein, mehrere Sätze pro Cluster zu selektieren.

10. **Zusammenfassung bilden:** Die selektierten Sätze (relevanten Sätze) werden nach Satzindex sortiert und zu einem Text zusammengefügt. Als Endprodukt entsteht eine extraktive Zusammenfassung.

11. **Evaluation:** Die generierten Zusammenfassungen werden mit Cosine Similarity mit dem extrahierten Abstract verglichen.

6.4.3 Resultate

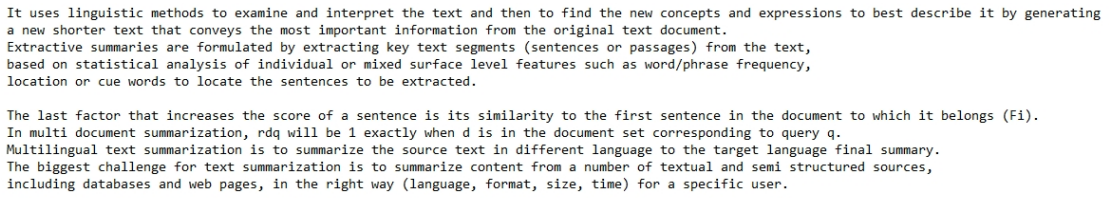


Abbildung 6.6: Beispiel einer generierten Zusammenfassung des Textes [2]

Cosine Similarity mit Abstract (Bester Wert)	n-Gram	Anzahl Cluster	Anzahl Sätze pro Cluster	Synonyme
0.59	1	3	2	nein
0.68	2	2	3	nein
0.66	2	2	3	ja
0.71	2	5	unterschiedlich	nein
0.61	2	5	unterschiedlich	ja

Tabelle 6.17: Automatische Zusammenfassung: Resultate

Für die Evaluierung wurden immer um die 50 Zusammenfassungen generiert, da je nach Startzentren die Cluster unterschiedlich sein können. In der Tabelle 6.17 stehen nur die besten Wertungen.

6.4.4 Fazit

Für das genaue Evaluieren des Algorithmus sind weitere Tests notwendig. Dies ist jedoch im Rahmen dieser Bachelorarbeit nicht möglich. Für die Evaluation wird davon ausgegangen, dass das Abstract die Elemente enthält, welche auch einen Text optimal zusammenfassen.

Kapitel 7

Schlussfolgerungen

Das Ziel der vorliegenden Bachelorarbeit war es, Wissen und technische Grundlagen im Bereich Text Mining aufzubereiten und das Erarbeitete in verschiedenen Aufgabenstellungen anzuwenden. Dabei wurde ein optimierter Algorithmus für die Textklassifikation entwickelt, welcher sowohl statistische als auch linguistische Aspekte beachtet. Für diese Aufgabe wurde die Open Source Software Rapidminer benutzt.
Der Algorithmus zur Klassifikation beinhaltet im Wesentlichen die folgenden Schritte:

1. Datenaufbereitung (Preprocessing)
2. Nachbearbeitung (Postprocessing)
3. Clustering
4. Klassifikation

Schritt für Schritt wurde ein Klassifikationssystem aufgebaut. Bei diesem System wurde der Wissensaufbau des Klassifizierers mithilfe eines Datensatzes vorgenommen. Mit Hilfe desselben Datensatzes wurde der Test zur Evaluation des Klassifizierers durchgeführt. Dabei konnten die einzelnen Datenaufbereitungsschritte

miteinander verglichen werden. Nebst dem Trennen der Dokumente in Fragmente (normalerweise Wörter, auch bekannt als Tokenizing) wurden zahlreiche Filter eingesetzt. Diese Filter ignorierten beispielsweise inhaltlich unwesentliche Wörter, sogenannte Stopwörter, aber auch Fragmente, welche eine bestimmte Länge über- oder unterschritten. Diese Methoden zogen zwar eine Verschlechterung der Vorhersage nach sich, es konnte jedoch die Anzahl der Lernparameter (Features) reduziert werden. Des Weiteren konnte beim verwendeten Datensatz und den gegebenen Klassen davon ausgegangen werden, dass genau diese gefilterten Fragmente keine Relevanz für den Klassifizierer haben dürften.
Weitere Schritte in Bezug auf die Datenaufbereitung waren hauptsächlich linguistischer Natur. Die Wörter wurden gruppiert nach Synonymen und Hyperonymen. Dieser Schritt war wichtig, zeigte es sich doch, dass die sprachlichen Aspekte mitberücksichtigt werden müssen, um eine Verbesserung der Vorhersage zu erreichen. Somit wurde in dieser Arbeit gezeigt, dass eine rein statistische Untersuchung eines Textes nicht genügt.
Die sprachliche Wichtigkeit zeigte sich unter anderem auch bei der Beachtung von Bigrams anstelle nur der Unigrams. Zwei Wörter, welche im Normalfall benachbart sind, gewinnen so an Bedeutung.
Nachdem verschiedene Datenaufbereitungsmöglichkeiten ausprobiert wurden und dieser Teil des Systems für ausgereift empfunden wurde, konnten die verschiedenen Klassifikationsalgorithmen verglichen werden. Dabei wurde vor allem mit den Algorithmen: ID3, Decision Tree, Naive Bayes und SVM gearbeitet. Das Testen ergab, dass sämtlich getestete Algorithmen etwa die gleiche Genauigkeit erreichten, wobei die Baumstrukturen (ID3 und Decision Tree) etwas bessere Werte ($\tilde{9}7\%$) erzielten. Der Klassifikationsalgorithmus scheint insofern für den verwendeten Datensatz unwichtig zu sein. Überraschenderweise versagten die Baumstrukturen bei einem einzelnen Test. Da jedoch eine genauere Untersuchung den Rahmen dieser Bachelorarbeit gesprengt hätte, konnte dieser Einzelfall nicht genauer untersucht werden.
Nach dem die Resultate der verschiedenen Klassifizierungsalgorithmen verglichen

worden sind, wurde versucht, mit einem grösseren Datensatz ein Klassifizierungsmodell aufzubauen. Letzteres sollte dann für die Bestimmung eines kleineren Datensatzes verwendet werden. Dabei wurde eine Genauigkeit von etwa 30% erreicht. Grund für diese schlechte Klassenzuweisung war auf ein Over Fitting zurückzuführen. Dies bedeutet, dass der Klassifizierer zu stark optimiert wurde, so dass er für die Trainingsdaten zwar bessere Vorhersagen traf, jedoch für einen anderen Datensatz unbrauchbar wurde. Ausserdem wurde für das Bestimmen des Klassifizierers zu lange Rechenzeiten benötigt ($\geq$ 2h).
Aus den obigen Erkenntnissen wurde ein finaler Prozess definiert, welcher anhand eines grossen Datensatzes instruiert wird, einen kleineren Datensatz zu klassieren. Dabei konnte eine Genauigkeit von rund 92.5% für zwei, 91.3% für drei und 83.5% für vier Klassen erreicht werden. Nebst diesen präzisen Voraussagen konnte auch die Berechnungszeit enorm verkürzt werden. Um dieses Ziel zu erreichen, wurden rund 100‘000 Features (2 Klassen zu je 1000 Dokumenten) auf 400 heruntergebrochen. Dies wurde durch die Anwendung einer Chi-Square Statistik ermöglicht. Dieses Verfahren ermöglicht, die Korrelation zwischen Feature und Klasse zu messen. Anhand der ermittelten Korrelation, konnte entschieden werden, welche Features für die Vorhersage relevant sind.

Weiterführend im Bereich Text Mining wurde dann das automatische Generieren von Zusammenfassungen untersucht. Da bei Rapidminer die Möglichkeit fehlte, den Ablauf genau zu steuern, wurde entschieden für diesen Teil Python mit dem Sprachpaket NLTK zu benutzen.

Mit Python konnte dann ein Prozess definiert werden, welcher zur Generierung von extraktiven Zusammenfassungen genutzt werden kann. Der Prozess kann wie folgt beschrieben werden:

1. Datenaufbereitung
2. Unterteilung in Sätze
3. Unterteilen in Wörter

4. Statistiken bilden
5. Berechnung der Distanzen zwischen Sätzen
6. Clusterbildung
7. Satz Selektierung
8. Zusammenfassung

Unter Datenaufbereitung wird in diesem Fall nur das Entfernen der nicht-strukturellen Satzelemente innerhalb des Dokumentes verstanden (beispielsweise können Formeln nicht zusammengefasst werden). Damit eine extraktive Zusammenfassung gelesen werden kann, ist es notwendig, dass Sätze extrahiert werden. Aus diesem Grund musste die erste Unterteilung auf Stufe Satz erfolgen. Damit jedoch die Sätze bewertet werden können, sind sie in die einzelnen Wörter zu zerlegen.
Eine der vermutlich wichtigsten Entscheidungen in diesem Bereich war es, die Sätze als einzelne Dokumente zu betrachten, auf welchen dann Wortstatistiken erstellt wurden.
Es wurde davon ausgegangen, dass ein Dokument verschiedene Themen behandelt. Daraus folgt, dass die Sätze zu gruppieren sind, so dass jedes Themengebiet in der Zusammenfassung repräsentiert wird. Um dies zu erreichen, wurde zuerst eine Tabelle aufgebaut, welche die Ähnlichkeit (nach Cosine Similarity) aller möglichen Satzpaare wiedergibt. Mit dieser Information konnten die Dokumente dann geclustert werden. Wieviele Cluster notwendig sind, hängt vom Dokument und von der Anzahl Themen ab, die es behandelt. Um den Rahmen dieser Bachelorarbeit nicht zu sprengen, konnte eine Veränderung der Clusterzahl nur beschränkt untersucht werden.
Nach der Gruppierung waren die wichtigsten Sätze innerhalb eines Clusters zu selektieren und diese dann in der richtigen Reihenfolge (so wie sie im Original Text auftreten) zusammenzufügen.
Durch diesen Algorithmus wurde eine Ähnlichkeit mit dem Abstract von rund 71% erreicht (Cosine Similarity).

7.1 Ausblick

Die vorgestellten Algorithmen könnten auf verschiedene Weise weiter optimiert werden.
Beispielsweise kam während der Bachelorarbeit die Idee auf, verschiedene Klassifikationsalgorithmen parallel laufen zu lassen. Texte, welche dann nicht immer der gleichen Klasse zugeordnet werden, würden als unbekannt gekennzeichnet. Dadurch könnte die Präzision der Vorhersage gesteigert werden.
Mit der neuen Version von Rapidminer werden zusätzlich Operatoren für Multi-Labelling angeboten. Diese Möglichkeiten, waren bei der Bearbeitung dieser Bachelorarbeit noch nicht vorhanden, würden jedoch interessante Aspekte nach sich ziehen.
Weitere Optimierungsmöglichkeiten gäbe es auch im Bereich „Named Entity Recognition“. Dabei werden Ausdrücke, welche eine tiefere Bedeutung haben, erkannt und mit Etiketten versehen (z.B. Werden Personennamen wie Barack Obama, mit weiteren Attributen wie „Person“, „American President“ assoziiert).

Bei der automatischen Zusammenfassung wäre es zudem sinnvoll gewesen, sich auf die Problematik zu konzentrieren, wie Themen, welche über mehrere Sätze hinweg erläutert werden, zusammengefasst werden können.

Kapitel 8

Persönlicher Bericht

Dadurch dass die Thematik recht weitläufig ist und es relativ schwer war, sie in dieser kurzen Zeit vollständig zu erfassen, war diese Bachelorarbeit gesamthaft eine Herausforderung. Ich konnte jedoch in dieser Zeit viel Wissen aneignen, welches mir bei zukünftigen Problemstellungen im Bereich Text Mining, aber auch sonst in der künstlichen Intelligenz von enormen Nutzen sein wird.
Durch die relativ freie Gestaltung, konnte ich gut die Herangehensweise einer wissenschaftlichen Arbeit erlernen, und zu jedem Zeitpunkt das Projekt in die Richtung steuern, die ich erkunden wollte. Dies erschwerte jedoch die Planung dieser Bachelorarbeit, so musste eine kurzfristige Planung verwendet werden.
Zusätzlich zu diesen Hürden waren für mich sämtlich verwendete Tools neu. So musste innerhalb kurzer Zeit Rapidminer und Python erlernt werden. Dies war eine zeitintensive Angelegenheit, jedoch eine wertvolle Erfahrung auch in Bezug auf zukünftige Arbeitsstellen.

Eine weitere Schwierigkeit, welche sich vor allem in der Dokumentation des Projektes zeigte, waren die zwei unterschiedlichen Aufgaben, Klassifikation und automatische Zusammenfassung. Jedoch konnte ich anhand diesen, die Parallelen sowie die Unterschiede zwischen verschiedenen Text Mining Systemen kennenlernen.
Auf Grund der in dieser Arbeit erlangten Erfahrungen, bin ich nun dazu in der Lage ein weiteres Projekt in diesem Bereich besser einzugrenzen, weiss wo sich

Schwierigkeiten befinden und wo nähere Untersuchungen nötig sind.
Am meisten profitierte ich von der Gegebenheit, dass diese Arbeit nicht nur theoretischer Natur war, sondern dass zusätzlich zu den erarbeiteten Grundlagen Algorithmen entwickelt wurden, welche auf diesen Grundlagen aufbauten. So konnte ich den Sinn von Text Mining erkennen und anhand meinen Aufgabenstellungen bemerken, was für ein Nutzen ein Benutzer durch die Text Mining Systeme erfährt.

Schlussendlich gilt zu sagen, dass eine Arbeit in diesem Bereich, zwar aufwändig ist, jedoch der Lerneffekt enorm war. Text Mining gehört zu einem Fachbereich, den ich auch in Zukunft verfolgen werde.

Literaturverzeichnis

[1] Hans Uszkoreiti. Vl einführung in die computerlinguistik. Website, year unknown. Available online at http://www.coli.uni-saarland.de/~hansu/Verarbeitung.html, visited on December 2nd 2013.

[2] Vishal Gupta; Gurpreet Singh Lehal. A survey of text summarization extractive techniques. In *Journal of emerging technologies in web intelligence*, volume 2, pages 258–268, 2010.

[3] Jiawei Han; Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2001. ISBN 1-55860-489-8.

[4] William J. Frawley; Gregory Piatetsky-Shapiro and Christopher J Matheus. Knowledge discovery in databases: An overview. In *AI Magazine*, volume 13, pages 57–70, 1992.

[5] Christopher D. Manning; Prabhakar Raghavan and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. ISBN 0521865719.

[6] Wu-Jun Li. Web search and mining: Lecture 1. Website - PPT, year unknown. Available online at http://www.cs.sjtu.edu.cn/~liwujun/course/wsm.html, visited on November 20th 2013.

[7] A. Anil Kumar and S. Chandrasekhar. Text data pre-processing and dimensionality reduction techniques for document clustering. In *International Journal of Engineering Research & Technology*, volume 1, 2012.

[8] Princeton. Naive bayes classifier. HTML. Available online at http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Naive_Bayes_classifier.html, visited on December 13th 2013.

[9] M. Porter. An algorithm for suffix stripping, 1980.

[10] A. Wong G. Salton and C.S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18:613–620, 1975.

[11] Charu C. Aggarwal and ChengXiang Zhai. *Mining Text Data*. Springer US, 2012. ISBN 978-1-4614-3222-7.

[12] Florian Markowetz. Klassifikation mit svm. PDF, 2003. Available online at http://lectures.molgen.mpg.de/statistik03/docs/Kapitel_16.pdf, visited on December 13th 2013.

[13] W. Konen. Naive bayes: Ein einfacher klassifikator. PDF, 2007. Available online at http://www.gm.fh-koeln.de/~konen/WPF-DM-Cup/04-Naive-Bayes.pdf, visited on December 13th 2013.

[14] Stanford University. Stanford log-linear part-of-speech tagger. HTML. Available online at http://nlp.stanford.edu/downloads/tagger.shtml, visited on December 13th 2013.

[15] Martin Lösch. Naive bayes- klassifikatoren. PDF. Available online at http://his.anthropomatik.kit.edu/users/loesch/LaborWissRepr-DHBW-KA-2011WS/downloads/Repetition-Naive_Bayesian_Networks.pdf, visited on December 13th 2013.

[16] Giorgio Ingargiola. Building classification models: Id3 and c4.5s. HTML. Available online at http://www.cis.temple.edu/~ingargio/cis587/readings/id3-c45.html, visited on December 13th 2013.