Treffer!

Automatische Textklassifizierung mit Hilfe des SVM-Verfahrens

Christa Zelch, Stephan Engel
Fachhochschule Augsburg
Diplomstudiengang Wirtschaftsinformatik/Informatik
E-Mail: chrzel@rz.fh-augsburg.de, engel@rz.fh-augsburg.de

2. Juli 2005

Kurzfassung: Die Kategorisierung aller Dokumente eines Unternehmens ist eine anspruchsvolle Aufgabe, die mit Hilfe intelligenter Sortierautomaten, so genannter automatischer Textklassifizierer, bereits zum großen Teil erledigt werden kann. Die folgende Arbeit soll einen Überblick über die verschiedenen Verfahren der automatischen Textklassifizierung vermitteln. Zunächst wird auf die Gemeinsamkeiten der Verfahren eingegangen und anschließend das regelbasierte und das statistische Verfahren erklärt. Auf den Mechanismus der Support Vector Machine (SVM) als dem wichtigsten Vertreter der statistischen Verfahren wird näher eingegangen. Ein praktischer Test des SVM-Algorithmus rundet die Arbeit ab, wobei die im Test entstandenen Ergebnisse präsentiert und interpretiert werden.

Schlüsselwörter: Textklassifizierung, SVM, Datenbanksysteme (Oracle 10g)

1 Einführung

Das geradezu exponentiell anwachsende Informationsaufkommen ist heutzutage eine Herausforderung, der sich große Unternehmen, wissenschaftliche Vereinigungen oder andere Interessenverbände stellen müssen. Gleichzeitig werden die Informationen durch die zunehmende Vernetzung für immer mehr Menschen zugänglich, unter anderem durch die Volltextsuche. Neben dieser spielt die automatische Textklassifizierung eine große Rolle. Ihre Aufgabe sind die inhaltliche Analyse von Texten und die Einteilung des gesamten entstehenden Contents in vordefinierte Klassen bzw. Kategorien, so dass anhand dieser Unterteilung relevante Informationen schneller gefunden werden können. Als Unternehmen ist es sinnvoll, eine automatische Textklassifizierungssoftware in ein Content Management System zu integrieren, da sie ein mächtiges Hilfsmittel zur Bändigung der Informationsflut darstellt.

Grundsätzlich lassen sich die Verfahren in regelbasierte und statistische Verfahren einteilen. Ziel dieser Arbeit ist es, diese Mechanismen vorzustellen. Insbesondere wird auf das SVM(Support Vector Machine)-Verfahren eingegangen und dieses anhand einer praktischen Untersuchung getestet. Die Analyse verwendet hierfür die Algorithmen, die in Oracle 10g integriert sind (vgl. [1], Kapitel 2.8.2). Die Wahl fiel unter anderem deshalb auf Oracle 10g, da es sich als schwierig erwies, (auch nur zu Testzwecken) frei verfügbare Klassifizierungssoftware zu erhalten, die den kompletten Klassifizierungsprozess unterstützt. Die Entscheidung für Oracle 10g hat sich (nicht nur mangels anderer Software) als sehr positiv herausgestellt. Die Treffergenauigkeit, die mit dem von Oracle verwendeten SVM-Algorithmus erzielt wurde, ist erstaunlich hoch. Die Beschreibung der unterschiedlich durchlaufenen Testfälle und die jeweiligen Ergebnisse werden am Ende dieser Arbeit präsentiert.

2 Die verschiedenen Verfahren im Überblick

Generell lassen sich Verfahren zur Textklassifizierung in regelbasierte und statistische Verfahren unterteilen. Während regelbasierte Verfahren ein zu klassifizierendes Dokument gegen einen Satz von Regeln prüfen und so die Klassenzugehörigkeit ermitteln, verlassen sich statistische Verfahren auf mathematische Modelle.

2.1 Gemeinsamkeiten

Allen Verfahren ist der Prozess gemein, der vor der eigentlichen Klassifizierung steht. Um einen Text klassifizieren zu können, müssen zunächst die einzelnen Wörter aus diesem Text extrahiert werden. Dieser Vorgang wird als „Lexen“ bezeichnet und muss auf jedes zu kategorisierende Dokument angewandt werden. Das Lexen kann unterschiedlich komplex ausfallen. Gute Verfahren reduzieren alle Flexionsformen auf den Wortstamm (Stemming), zerlegen Komposita (Tokenizing) und vernachlässigen aussageschwache Wörter (Stoppwörter), zum Beispiel Bindewörter und Präpositionen. Als Ergebnis erhält man alle relevanten Wörter des zu kategorisierenden Dokuments. Diese Wortmenge wird dann an den Klassifizierungsalgorithmus übergeben.

2.2 Regelbasiertes Verfahren

Regelbasierte Verfahren arbeiten zumeist mit einem Satz manuell erstellter Regeln. Zwar gibt es auch die Möglichkeit, sich einen Regelsatz aus einer Trainingsmenge automatisch generieren zu lassen, am Klassifizierungs-verfahren ändert sich hierdurch jedoch nichts. Regelbasierte Verfahren basieren auf den booleschen Operatoren. Damit ein Dokument einer Kategorie zugeordnet wird, muss es die Regeln für diese Kategorie erfüllen. Die Regeln einer Kategorie sind untereinander wiederum mit booleschen Operatoren verknüpft. Damit wird festgelegt, ob ein Dokument beispielsweise alle Regeln erfüllen muss (UND-Verknüpfung) oder nur eine der Regeln (ODER-Verknüpfung).

Der Vorteil regelbasierter Verfahren ist, dass eine hohe Treffergenauigkeit erzielt wird, wenn wenige, eindeutige und simple Regeln benutzt werden. Die Treffergenauigkeit ist definiert als die Qualität und Menge der korrekt zugeordneten Dokumente. Ein weiterer Vorteil der regelbasierten Verfahren ist ihre einfache Handhabung. So werden hierbei zum Beispiel keine Trainingsdokumente benötigt und das Verfahren kann mit wenig Aufwand selbstständig implementiert werden.

Ein Nachteil des Verfahrens ist, dass ein Dokument entweder einer Klasse zugeordnet wird oder nicht. Dieses Problem kann jedoch durch Gewichtung der Regeln behoben werden. Dies bedeutet, dass jeder Regel ein Zahlenwert (Gewicht) zugewiesen wird. Jeder Klasse wird wiederum ein Schwellwert zugeteilt. Die Summe der Gewichte aller zutreffenden Regeln einer Kategorie wird mit dem Schwellwert der Klasse verglichen. Ist das Gesamtgewicht größer als der Schwellwert, wird das Dokument dieser Klasse zugeordnet. Durch die Abweichung des Gewichts vom Schwellwert kann außerdem eine Aussage darüber getroffen werden, wie genau ein zu klassifizierender Text in eine Kategorie passt.

Ein anderer Nachteil regelbasierter Verfahren ist, dass mit zunehmender Anzahl und Komplexität der Klassen die Prozedur der Regelfindung schwierig bis unmöglich ist (vgl. [2]).

2.3 Statistisches Verfahren

Die verschiedenen statistischen Verfahren arbeiten zum Teil mit zueinander ähnlichen Algorithmen und setzen meist auf Vektormodelle. Die zugrunde liegende Idee ist relativ simpel. Jeder Text wird als Vektor repräsentiert. Jedes relevante Wort entspricht einer Dimension im Vektor. Alle Vektoren haben die gleiche Dimensionalität. Jedem relevanten Wort wird eine Zeile (Dimension) in diesem Vektor zugeordnet. Der zugehörige Wert beschreibt, wie häufig das ensprechende Wort im Text vorkommt. Die Zuordung der Wörter zu den Dimensionen ist für alle Texte gleich. Wird beispielsweise im ersten Text die Zuordnung $\begin{matrix} Hund \\ bellt \end{matrix} = \overrightarrow{x} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ gewählt, so muss im zweiten Text der ersten Zeile des Vektors ebenfalls das Wort „Hund“ zugeordnet werden. Sollen neue Wörter berücksichtigt werden, muss die Dimensionalität des Vektors erhöht werden. Dies

ist wichtig, damit sich die Richtung der Vektoren anhand der ihnen zugrunde liegenden Wörter unterscheidet. Würde man jedes Dokument einfach in einen Vektor umwandeln, ohne die Wörter aus anderen Dokumenten zu berücksichtigen, so hätten völlig unterschiedliche Dokumente die gleichen Vektoren, was eine Unterscheidung unmöglich machen würde. Außerdem können einzelne Wörter stärker bewertet werden.

An einem einfachen Beispiel soll der komplette Vorgang veranschaulicht werden: Der Satz: „Der Hund bellt wie ein Hund“würde so zunächst in einen Vektor der Form: $\overrightarrow{x} = \begin{pmatrix} 1 \\ 2 \\ 1 \\ 1 \\ 1 \end{pmatrix}$ umgewandelt werden. Muss nun ein weiteres Dokument, zum Beispiel der Satz„Der Hund bellt die ganze Zeit“, als Vektor dargestellt werden, so wird wie zuvor beschrieben die Dimensionaltiät aller Vektoren erhöht. Idealerweise wurden zuvor inhaltslose Wörter entfernt und alle Wörter zudem auf ihre Wortstämme reduziert (Stemming, siehe 2.1). Im oben genannten Beispiel würden sich dann folgende Vektoren ergeben: $\begin{matrix} Hund \\ bellen \\ Zeit \end{matrix} = \overrightarrow{x} = \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}$ und $\begin{matrix} Hund \\ bellen \\ Zeit \end{matrix} = \overrightarrow{y} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$

Typische Verfahren zur Wortgewichtung sind die Bewertung der Termfrequenz (TF) und der inverse Dokumentfrequenz (IDF).Die Termfrequenz gibt an, wie oft ein Wort in einem Dokument vorkommt; die Dokumentfrequenz, in wie vielen Dokumenten das Wort ein- oder mehrmals auftaucht. Je häufiger ein Wort in einem Text erscheint, umso relevanter ist es für die Klassifizierung. Findet sich ein Wort aber in sehr vielen Dokumenten und Klassen, dann sinkt seine Bedeutung für die Klassifizierung. Dies spiegelt sich in der TF-IDF-Formel(vgl. [2]) wider:

$w_{ij} = t_{ij} \cdot log\left(\frac{N}{f_j}\right)$ mit
w_{ij} = das Gewicht des Terms (Wort) j in Dokument i
t_{ij} = Zahl des Auftretens von Term j in Dokument i
N = gesamte Anzahl der Dokumente (über alle Klassen)
f_j = Zahl der Dokumente, die den Term enthalten j enthalten

Wird beispielsweise festgestellt, dass die Wörter „Hund“ und „bellen“ in der Kategorie „Tiere “ häufig vorkommen, so kann das Gewicht dieser Wörter im Vektor erhöht werden, indem ihr Wert zum Beispiel mit drei multipliziert wird. Man erhält:

$\begin{matrix} Hund \\ bellen \\ Zeit \end{matrix} = \overrightarrow{x} = \begin{pmatrix} 6 \\ 3 \\ 0 \end{pmatrix}$ und $\begin{matrix} Hund \\ bellen \\ Zeit \end{matrix} = \overrightarrow{y} = \begin{pmatrix} 3 \\ 3 \\ 3 \end{pmatrix}$

Diese Beispiele sind stark vereinfacht. Eine wirkliche, aber lösbare Herausforderung für die Klassifizierungssoftware stellen die 10.000 Reuters-Nachrichten aus den Jahren 1987-1991 dar, die oft als Benchmark für verschiedene Klassifizierungsalgorithmen dienen.

Bekannte statistische Algorithmen sind zum Beispiel das Rocchio-Verfahren, k-Nearst Neighbour (kNN) und der SVM (Support Vector Machine) Algorithmus.(vgl. [2]) Die statistische Evaluierung für die vorliegende Arbeit wurde mit Hilfe des SVM-Algorithmus durchgeführt, da dieser von Oracle 10g unterstützt wird und zudem eine hohe Trefferquote (bis zu 90%) durchaus ermöglicht.

3 Das Support-Vector-Machine-Verfahren (SVM)

Das SVM-Verfahren ist ein lernendes statistisches Verfahren zur automatischen Textklassifizierung. Der Algorithmus, der den Text klassifizieren soll, lernt eine Klassifizierungsfunktion anhand von Trainingsdokumenten. Diese Trainingsdokumente sind bereits in Kategorien eingeteilt und werden bereitgestellt.

Nachdem die Trainingsdokumente für die verschiedenen Klassen dem System bekannt gemacht worden sind, bildet dieses die Dokumentvektoren. Danach berechnet das System eine Hyperebene, das heißt eine Ebene, die so zwischen den Vektoren der verschiedenen Klassen liegt, dass der Abstand zu allen Klassen maximal ist. Im vereinfachten zweidimensionalen Beispiel sieht das aus wie in Abbildung 1

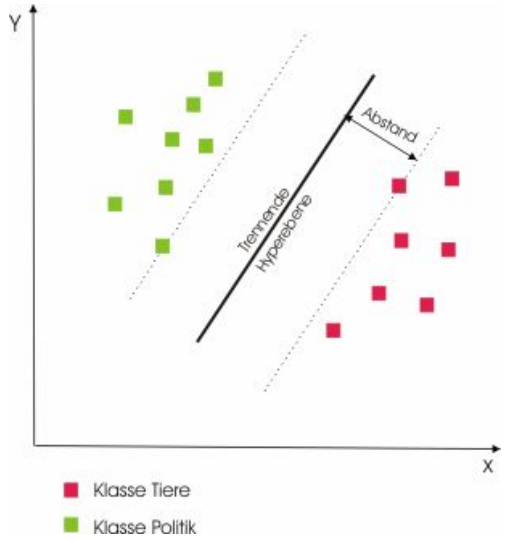

Abbildung 1: *Trennung der Dokumentvektoren durch die Hyperebene (vgl. [3])*

Das Problem hierbei ist, dass die guten und die schlechten Beispiele keineswegs immer so einfach vorliegen. Wahrscheinlich liegen die Vektoren der unterschiedlichen Klassen ganz anders zueinander im Raum, möglicherweise wie in Abbildung 2. Hier kann die Berechnung der Hy-

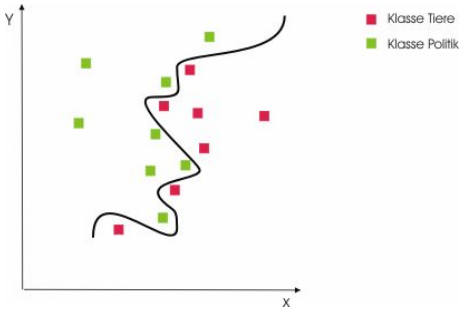

Abbildung 2: *Erschwerte Trennung der Dokumentvektoren*

perebene sehr kompliziert und in höheren Dimensionen sogar unmöglich werden. Der Trick, der nun angewandt wird, ist der, dass die Hyperebene für eine Menge n-dimensionaler Vektoren in einer Dimension größer n leichter berechnet werden kann, wie Abbildung 3veranschaulicht. gut veranschaulicht. Die Trennung der beiden Mengen ist im zweidimensionalen Raum nur mit Hilfe

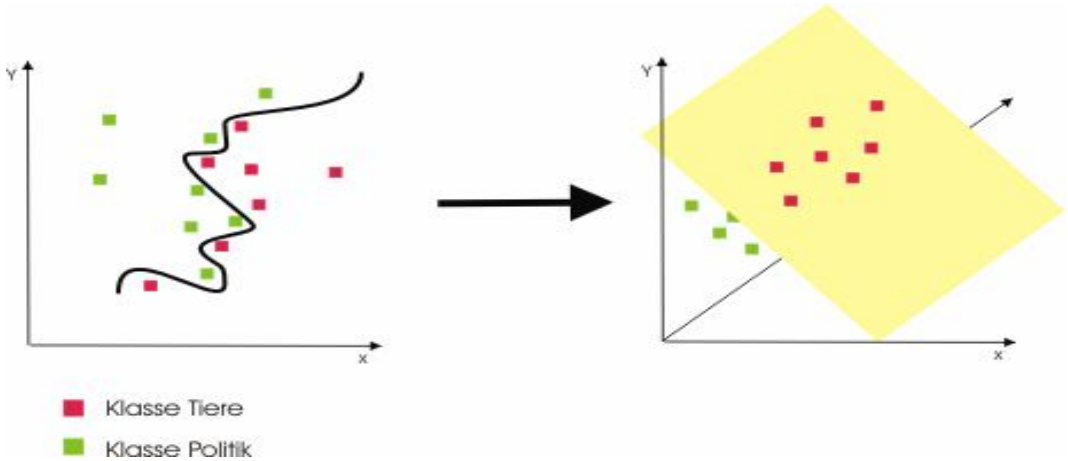

Abbildung 3: *Eine Erweiterung des Raumes um eine weitere Dimension erleichtert die Trennung*

einer schlagenförmigen Linie möglich. Im dreidimensionalen Raum hingegen reicht eine normale gerade Ebene.

Zwar gibt es auch hier Probleme bei der Berechnung; diese können aber mit Hilfe des so genannten „Kerneltricks“ umgangen werden. Auf die mathematischen Details soll hier nicht weiter eingegangen werden. Ist die Ebene berechnet, muss nur noch der Support Vector für jede Klasse berechnet werden. Der Support Vector (Stützvektor) ist der Vektor, der senkrecht auf der Hyperebene steht und auf das der Ebene am nächsten gelegene Dokument zeigt. Diese Vektoren müssen vom System gespeichert werden (vgl. [3]).

Ist ein Dokument neu zu klassifizieren, so wird zunächst sein Vektor berechnet. Der Abstand dieses Vektors zu den Stützvektoren gibt an, wie gut ein Dokument in eine Klasse passt, und ist sozusagen die Regel, mit der ein Dokument getestet wird. Die Abstandsberechnung kann über den Kosinus des Winkels dieser beiden Vektoren berechnet werden.

Der Vorteil des SVM-Algorithmus ist einmal, dass Overfitting verhindert wird, da die Komplexität der Klassen mit in den Trainingsalgorithmus einfließt. Beim Overfitting ist die Regel zu komplex und es werden nur noch Dokumente als passend erkannt, die der Trainingsmenge entnommen sind, wie beispielsweise in Abbildung4. Der SVM zeichnet sich weiterhin als sehr per-

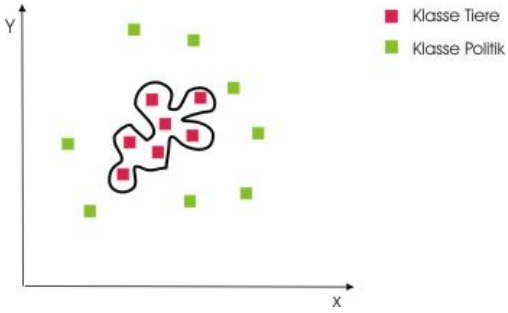

Abbildung 4: *Overfitting Effekt: Die Klassen werden zu genau getrennt. Es ist schwierig, Dokumente der Klasse Tiere zuzuordnen.*

formant aus, da pro Klasse nur eine Regel (der Abstand zum Stützvektor) geprüft werden muss.

4 Praktische Analyse des SVM-Klassifizierers

Die Untersuchung des SVM-Klassifizierers auf seine praktische Tauglichkeit wird im Folgenden beschrieben. Als Implementierung diente hierfür der in Oracle 10g integrierte Algorithmus.

4.1 Vorbereitungen

Zuerst war ein adäquates Datenbankschema zu erstellen. Dies besteht aus drei Teilen:

Dokumententabellen

Insgesamt wurden fünf Dokumententabellen angelegt. Dies war erforderlich, um verschiedenartige Testfälle abzudecken, wie zum Beispiel die Klassifizierung mit bzw. ohne Wortstammreduktion oder die Verwendung einer variierenden Anzahl von Stoppwörtern. Entsprechende Parameter wurden dem Klassifizierungsalgorithmus über den Volltextindex, der auf den Trainingsdokumenten liegt, übergeben. Da pro Tabellenspalte nur ein Volltextindex angelegt werden kann, sind mehrere Dokumententabellen oder zumindest mehrere Spalten notwendig, um die gewünschten Testfälle abzudecken. Es wurden folgende Indexe angelegt:

- dokumente_idx: Defaulttextindex (keine Stoppliste)
- dokumente_1_idx: Defaultindex mit aktiviertem Stemming
- dokumente_2_idx: Stoppliste mit 100 Wörtern und Stemming
- dokumente_3_idx: Stoppliste mit 1.000 Wörtern und Stemming
- dokumente_4_idx: Stoppliste mit 4.095 Wörtern und Stemming

Zuordnungstabellen

Um zu testen, wie sich eine unterschiedliche Anzahl von Klassen auf den Algorithmus auswirkt, wurden drei Zuordnungstabellen angelegt. In den Zuordnungstabellen wird festgelegt welches Trainingsdokument zu welcher Klasse gehört. In einer Tabelle wurden sämtliche Kategorien verwendet, in der zweiten Tabelle fünf Kategorien und in der letzten zwei Kategorien. Da der Algorithmus nur solche Dokumente zum Training verwendet, die in der jeweiligen Zuordnungstabelle aufgeführt sind, konnten wir alle fünf Dokumententabellen in Kombination mit jeder Zuordnungstabelle verwenden. Dadurch wurde eine große Trainingsflexibiliät erreicht.

Kategorientabelle

Die Kategorientabelle definiert neun Kategorien. Sie wäre eigentlich nicht notwendig, da für den Algorithmus lediglich die Zuordnung der Dokumente zu den Kategorien relevant ist. Die Kategorientabelle dient lediglich als Look-up-Tabelle.

Weiterhin galt es, genügend Trainingsdokumente bereitzustellen. Dies stellte eine gewisse Herausforderung dar, da für aussagekräftige Ergebnisse mindestens 50 Dokumente pro Klasse benötigt werden. Aufgrund dieser Anforderung haben wir uns dafür entschieden, auf eine Sammlung von c't-Ausgaben im HTML-Format zurückzugreifen. Diese haben den Vorteil, dass sie bereits in unterschiedliche Kategorien unterteilt sind. Es galt also nur, die Dokumente in die von uns vordefinierten Kategorien einzusortieren und diese dann in die Datenbank zu laden. Unsere Kategorien sind stark an die c'T Kategorien angelehnt.

Um das manuelle Klasifizieren der Dokumente zu ermöglichen, mussten sie lokal vorliegen. Deshalb kopierten wir mit Hilfe des Programms „winscp“ alle c'T Artikel aus den Jahren 2002, 2003 und 2004 auf unsere Festplatte. Mit Hilfe eines von uns in Python geschriebenen Skripts wurden diese Artikel in neun Kategorien einsortiert:

– Ausbildung

- DVD-Brenner
- Forschung
- Kamera
- Netzwerk
- Notebook
- Prozessor
- Recht
- Scanner

Diese Vorgehensweise war möglich, da im „<title>“ Tag der HTML-Dateien die im c’t Magazin verwendeten Kategorien festgehalten sind. Das Python-Skript überprüft mit Hilfe regulärer Ausdrücke, ob ein bestimmter String in diesem Tag vorkommt, und sortiert im Erfolgsfall das HTML-Dokument in die jeweilige Kategorie. Die Kategorien werden durch Ordner repräsentiert.

Die so kategorisierten Artikel aus den beiden Jahren 2003 und 2004 wurden danach mit Hilfe eines in Java geschriebenen Programms in die Datenbank geladen. Außerdem trugen wir in die Kategorientabelle ein, welchen Klassen die Trainingsdokumente zugeordnet sind.

Um die Auswirkung von Stoppwörtern auf die Klassifizierung überprüfen zu können, wurden anschließend noch drei Stoppwortlisten mit jeweils 100, 1.000 und 10.000 Stoppwörtern in die Datenbank geladen. Bei der dritten Liste brach der Import nach 4.095 Wörtern jedoch ab. Der Grund dafür war, dass Oracle maximal diese Anzahl an Stoppwörtern pro Liste zur Verfügung stellt. Die Stoppwortlisten bezogen wir über [4]. Wie auf der Internetseite vermerkt, können sich auch andere Reihenfolgen bei der Häufigkeit der Wörter ergeben. Es ist wichtig, dass bei der Auswahl der Stoppwortlisten auf eine gute Qualität dieser geachtet wird. Die Qualität der von uns verwendeten Stoppwortlisten lässt ab einer gewissen Anzahl an Wörtern leider etwas nach, da auch eigentlich relevante Wörter wie beispielsweise “Universität „ in ihnen enthalten sind.

Ferner wurden jeweils zwei Artikel pro Kategorie aus dem Jahr 2002 zum späteren Testen des Algorithmus in eine weitere Tabelle der Datenbank geladen. Diese Dokumente sind die eigentlich zu klassifizierenden Dokumente.

Zuletzt wurden fünf Präferenzen für den Trainingsprozess angelegt. Diese geben an, wie viele Wörter der Trainingsprozess maximal pro Dokument berücksichtigen soll, wie viele Wörter insgesamt in den Trainingsprozess einfließen und ob die Wortstämme, die vorhandenen Flexionsformen oder die Kombination aus beiden als einzelne Wörter gezählt werden sollen. Die fünf Präferenzen haben folgende Eigenschaften:

defaultsvmpref
: Diese Präferenz verwendet die Standardeinstellungen des Oracle-Systems. Die maximale Wortzahl pro Dokument beträgt 50, die Gesamtanzahl an Wörtern ist 3.000 und es werden die Flexionsformen indiziert (kein Stemming). Auf die vorherigen Terminologien übertragen bedeutet dies, dass die maximale Vektorgröße 3000 Zeilen beträgt und pro Dokument maximal 50 Wörter in den Vektor einfließen.

pref1
: Bei dieser Präferenz haben wir die maximale Wortanzahl pro Dokument auf 500 und die Gesamtwortanzahl auf 45.000 erhöht. Auch hier ist Stemming nicht aktiviert.

pref2
: Hier wurde zusätzlich zu der erhöhten Wortanzahl Stemming eingeschaltet, wobei die Flexionsformen nicht mehr berücksichtigt werden.

pref3
Im Gegensatz zu pref2 wurde hier zusätzlich die Kombination aus Flexionsformen und Wortstammreduktion benutzt.

pref4
In der letzten Präferenz wurde die Defaulteinstellung dahingehend verändert, dass wie in pref2 Stemming aktiviert und Flexionsformen deaktiviert wurden. Die Wortanzahlen entsprechen denen der Defaulteinstellung.

4.2 Trainingsprozess

Nachdem alle Vorbereitungen getroffen worden waren, konnten die einzelnen Testfälle definiert werden. Ein Testfall setzt sich zusammen aus der Kombination von Dokumententabelle, Zuordnungstabelle und SVM-Präferenz. Insgesamt hat der Algorithmus 13 von uns definierte Testfälle durchlaufen, wobei sich die Testfälle in fünf verschiedene Gruppen aufteilen lassen, die bestimmte Fragen stellen.

Testgruppe 1: Wie wirkt sich eine unterschiedliche Anzahl an Kategorien aus?
Indem die Kategorienanzahl mit Hilfe der Zuordnungstabellen variiert, aber immer der gleiche Textindex und die gleiche SVM-Präferenz verwendet wurden, war zu evaluieren, ob der Algorithmus erkennbare Stärken oder Schwächen bei großer Kategorienanzahl aufweisen würde.

Testgruppe 2: Wie wirkt sich eine unterschiedliche Anzahl an Stoppwörtern aus?
Mit den drei verschieden langen Stoppwortlisten sollte analysiert werden, wie sich Stoppwörter auf die automatische Klassifikation auswirken würden. Die Testfälle variieren folglich die Dokumententabellen; SVM-Präferenz und Zuordnungstabelle bleiben dagegen gleich.

Testgruppe 3: Wie wirken sich Stoppwörter bei weniger Kategorien aus?
Nach dem vorhergehenden Testfall sollte dann analysiert werden, ob sich die Ergebnisse auch auf ein Szenario mit weniger Kategorien übertragen lassen würden.

Testgruppe 4: Wie wirkt sich Stemming aus?
Bei diesem Testfall sollte die Wortstammreduktion getestet werden. Die Testfälle wurden so gewählt, dass der Textindex der ausgewählten Dokumententabelle mit 100 Stoppwörtern angelegt wurde. Durch die kurze Stoppwortliste wurde sichergestellt, dass nicht zu viele, möglicherweise relevante, Wörter aus dem Raster fallen. Zudem wurde analysiert, wie sich Stemming in Kombination mit einer vom Algorithmus unterschiedlich zu berücksichtigenden Wörteranzahl auswirken würde.

Testgruppe 5: Wie wirkt sich eine unterschiedliche Anzahl berücksichtigter Wörter aus?

Dieser Testfall will klären, ob es für den Algorithmus sinnvoll ist, mehr oder aber weniger Wörter bei einem Trainingsprozess zu berücksichtigen. Um die Zahl der zu berücksichtigenden Wörter zu erhöhen, wurden sowohl Flexionsformen als auch Wortstämme verwendet.

Für jeden der Testfälle haben wir eine eigene Ergebnistabelle angelegt. In diesen Tabellen speichert der Algorithmus beim jeweiligen Trainingsprozess sein Ergebnis ab. Um eine Klassifizierung zu ermöglichen, musste zudem über die Ergebnisse ein Regelindex gelegt werden. Dies ist erforderlich, da der

`matches()`

Operator des Oracle-Systems diesen Index benötigt, um analog zum Trainingsprozess – der seine Textpräferenzen aus dem Volltextindex bezieht – seine Textpräferenzen aus diesem Index zu generieren. Der

`matches()`

Operator benötigt diese Einstellungen, damit er den Text nach den selben Verfahren in einen Vektor umwandleln kann, wie bei den Trainingsdokumenten. Würden hier andere Einstellungen verwendet werden, würde dies das Ergebnis verfälschen.

4.3 Ergebnisse

Als nächstes wurde der Klassifizierungsprozess gestartet. Der Algorithmus durchlief die 13 unterschiedlichen Testfälle mit den 18 zu klassifizierenden Dokumenten. Die Ergebnisse werden im Folgenden aufgeführt und interpretiert.

Testgruppe 1: Die unterschiedliche Anzahl an Kategorien hat kaum Auswirkungen auf die Güte des Ergebnisses. Erhöhten wir die Kategorienzahl von zwei auf fünf, so war nur bei einem Text eine Abweichung in der Treffergenauigkeit festzustellen. Interessanterweise verschlechterte sich die Trefferwahrscheinlichkeit nicht wie ursprünglich angenommen, sondern verbesserte sich sogar. Dieses Verhalten können wir nicht erklären.

Steigerten wir die Kategorienanzahl auf zehn, so war eine marginale Verschlechterung der Trefferwahrscheinlichkeit festzustellen, durchschnittlich um drei Prozent.

Testgruppe 2: Die unterschiedliche Anzahl an Stoppwörtern wirkt sich zunächst anscheinend kaum auf das Klassifizierungsergebnis aus. Bei näherer Betrachtung fallen allerdings gewisse Tendenzen auf: zwar ist die Reihenfolge der Ergebnisse über die verschiedenen Testklassen hinweg mehr oder weniger gleich verteilt, aber weniger Stoppwörter wirken sich insgesamt positiv aus.

Sollen jedoch lange und inhaltlich stark fachbezogene Texte klassifiziert werden, wirken sich lange Stoppwortlisten positiv auf das Ergebnis aus. Dies erklärt sich dadurch, dass bei solchen Texten durch große Stoppwortlisten die kategorienspezifischen Wörter stark an Bedeutung gewinnen, die auf keiner Stoppwortliste zu finden sind.

Testgruppe 3: Um zu überprüfen, ob sich das zuvor erzielte Ergebnis in Bezug auf Stoppwortlisten auch auf eine geringere Anzahl an Kategorien übertragen lässt, wiederholten wir den Test mit nur zwei Kategorien. Dabei fiel uns auf, dass das vorherige Ergebnis auch in diesem Testfall fast unverändert blieb. Das heißt, bei der Auswahl der Stoppwortliste spielt es keine Rolle, wie viele Kategorien verwendet werden.

Testgruppe 4: Stemming wirkt sich zunächst negativ auf das Testergebnis aus. Allerdings fanden wir heraus, dass bei sehr kurzen fachspezifischen Texten die Deaktivierung von Stemming die Qualität des Ergebnisses sehr stark verschlechtert. Weiterhin beobachteten wir, dass bei aktiviertem Stemming auch die zu berücksichtigende Wortanzahl erhöht werden sollte, da damit nahezu durchgehend bessere Ergebnisse erzielt wurden als bei Stemming mit wenig berücksichtigten Wörtern. Die Ergebnisse entsprachen dann der guten Qualität der Ergebnisse ohne Stemming. Unsere Empfehlung ist demzufolge, generell auf Stemming zu verzichten, wobei die aufgeführten Ausnahmen zu berücksichtigen sind.

Testgruppe 5: Eine große zu berücksichtigende Wortanzahl wirkt sich bei gut differenzierten Kategorien kaum aus. Dies erscheint logisch, da auch bei geringer Wortanzahl die aussagekräftigen Wörter einer Kategorie stark ins Gewicht fallen. Eine Erhöhung der Wortanzahl führt lediglich dazu, dass mehr Wörter mit berücksichtigt werden, diese aber für die einzelnen Klassen kaum

Relevanz besitzen. Überraschend wird es, wenn kurze, inhaltlich stark variierende Texte klassifiziert werden sollen. In diesem Fall wirkt sich eine größere Anzahl an zu berücksichtigenden Wörtern positiv aus, da keine entscheidenden Wörter aus dem Raster fallen.

Wenn der Algorithmus sowohl Wortstamm als auch Flexionsform berücksichtigen soll, wirkt sich das in dieser Testgruppe günstig aus. Wenn mit einer großen Anzahl an Wörtern gearbeitet wird, sollte man beide Features (Wortstamm und Stemming) aktivieren. Ein negativer Effekt war jedenfalls in keinem Testfall zu beobachten.

5 Fazit

Diese Arbeit hat verschiedene Methoden der automatischen Textklassifizierung am Beispiel von Computertexten vorgestellt.

Das regelbasierte Verfahren wurde theoretisch beschrieben und das statistische Verfahren durch einen praktisch durchgeführten Klassifikationstest mit Hilfe des SVM-Algorithmus der Oracle Datenbank 10g untermauert. Dieser hat seine Qualität durch eine sehr hohe Treffergenauigkeit – selbst bei ausgeprägter Verschiedenartigkeit der Testfälle – bewiesen. Hierbei variierte nicht nur die Kategorien- und Stoppwortanzahl, sondern es wurde auch mit aktiviertem bzw. deaktiviertem Stemming getestet.

Beim Klassifizieren fiel auf, dass mit den Defaulteinstellungen im Regelfall sehr gute Ergebnisse zu erzielen sind. Unbedingt sinnvoll erscheint, eine Stoppwortliste zu benutzen, wobei hier gilt: weniger ist mehr. Wie viele Stoppwörter genau verwendet werden sollten, ist im jeweiligen Testlauf zu evaluieren. Der nötige Aufwand ist nicht allzu groß, so dass mehrere Tests machbar erscheinen.

Der Algorithmus ermöglicht eine äußerst zuverlässige Kategorisierung, ohne auf einen allzu umfangreichen Regelsatz angewiesen zu sein, was vor allem aus Performancegründen für diese Art der Klassifizierung spricht.

Abschließend kann man sagen, dass der SVM-Algorithmus als Repräsentant des statistischen Verfahrens für das Content Management mit dessen vielfältigen Facetten bestens gerüstet ist.

Literatur

[1] Oracle Text Refeference
`http://download-west.oracle.com/docs/cd/B14117_01/text.101/b10730/toc.htm`

[2] Thomas Brückner, Holger Dambeck: Sortierautomatten
c't Magazin für Computertechnik, Ausgabe 19/03 Seite 194

[3] Florian Markowetz: Klassifikation mit Support Vector Machines
`http://lectures.molgen.mpg.de/statistik/docs/Kapitel_16.pdf`

[4] Stoppwortlisten der Uni Leipzig
`http://wortschatz.uni-leipzig.de/html/wliste.html`