Köln, den 25. April 2015

Studiengang Informationsverarbeitung
Wintersemester 2014/2015
Sprachliche Informationsverarbeitung
Hauptseminar: „ Linguistic Software Engineering“
bei Prof. Dr. Jürgen Rolshoven

Textklassifikation mit Support Vector Machines

vorgelegt von:
Alena Tabea Geduldig

Inhaltsverzeichnis

1 Textklassifikation

Durch die immense und immer weiter zunehmende Zahl an online verfügbaren Texten, wächst auch das Interesse daran, den Informationsgewinn aus digitalen Texten zu automatisieren. Eine der Hauptdisziplinen des sogenannten *Data Minings* ist die automatische Textklassifikation, also die inhaltsbasierte Klassifikation natürlichsprachlicher Texte in vordefinierte Kategorien. Ein klassisches Beispiel für Textklassifikationssysteme sind die Spam-Filter-Programme von E-Mail-Diensten. Eingehende E-Mails werden auf Grundlage ihrer Inhalte in die Kategorien Spam und nicht-Spam unterteilt. Auch eine Klassifikation in mehrere Kategorien ist möglich, zum Beispiel die Einordnung von Nachrichtenartikeln in verschiedene Zeitungsrubriken. Ein entscheidender Vorteil von auf diese Weise vorklassifizierten Textbeständen ist der erleichterte Zugang zu gewünschten Informationen (vgl. Brückner 2004 S. 496). So kann eine vorgeschaltete Klassifikation beispielsweise den Suchraum für eine automatische Informationsextraktion oder Question-Answering Systeme verkleinern.

Zur Klassifikation von Texten existieren eine Reihe unterschiedlicher Verfahren (vgl. Sebastiani 2002), die sich grob in zwei Hauptrichtungen unterteilen lassen: Regelbasierte Verfahren und maschinelle Lernverfahren (ML). Regelbasierte Verfahren beruhen auf der Abarbeitung strikter Regeln für jede Klasse, die üblicherweise von einem Anwender manuell codiert werden. Solche Verfahren arbeiten sehr präzise. Ihr Nachteil liegt vor allem in der aufwändigen Regelerstellung, die zudem gemeinhin domänenspezifisch und damit nicht übertragbar auf andere Klassifikationsprobleme ist. In den letzten Jahren ging der Trend daher zu maschinellen Lernverfahren. ML-basierte Verfahren beruhen auf dem Prinzip ‚Lernen durch Beispiele'. Anhand eines repräsentativen, vorausgezeichneten Trainingskorpus werden Klassenprofile (sogenannte Modelle) für jede Klasse gebildet, auf deren Grundlage ein Klassifikator Zuordnungen neuer (d.h. noch nicht ausgezeichneter) Daten vornimmt. Eine der derzeit vielversprechendsten Methoden zur Textklassifikation ist die Verwendung sogenannter Support Vector Machines, welche 1995 vom Mathematiker Wladimir Wapnik vorgestellt wurden (vgl. Wapnik 1995).

1.1 Der Job Ad Section Classifier (JASC)

Im Zuge eines Kooperationsprojektes vom Bundesinstitut für Bildung (BIBB) und der Sprachlichen Informationsverarbeitung der Universität zu Köln wurde im vergangenen Jahr ein Softwareframework zur automatischen Klassifikation von Abschnitten aus Stellenanzeigen entwickelt (vgl. Hermes 2015). Das Ziel dieses Framework war es, verschiedene

Textklassifikationsverfahren gegeneinander evaluieren zu können und die für diese Problemstellung geeigneten Methoden und Vorverarbeitungsschritte zu ermitteln. Einzelne Abschnitte aus Stellenanzeigen wurden hierfür in vier inhaltsmotivierte Kategorien klassifiziert. Die Definition der vier Klassen begründete sich aus einem beobachtbaren spezifischen Schema, auf das sich die Stellenanzeigen zurückführen ließen. So ließen sich die darin enthaltenen Informationen in die Kategorien

1. Arbeitgeber-Information
2. Job-Information
3. Anforderungen an den Bewerber und
4. Formalia/Sonstiges

aufteilen. Abbildung 1.1 zeigt die Zuordnung einer anonymisierten Stellenanzeige in die genannten Kategorien.

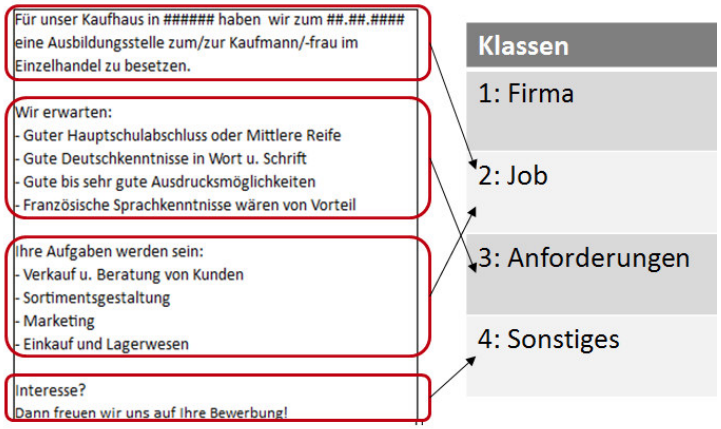

Abbildung 1.1: Zuordnung von Abschnitten einer anonymisierten Stellenanzeige in inhaltlich vordefinierte Klassen (Quelle: Hermes 2015, S. 3)

Diese Art der Klassifikation wird in der Literatur häufig mit dem Begriff **Zone-Analysis** bezeichnet. Gemeint ist damit die Klassifikation einzelner Abschnitte (Zones) aus Texten anstelle der Klassifikation vollständiger Dokumente. Zone-Analysis wird häufig zur Verarbeitung wissenschaftlicher Aufsätze oder Zeitungsartikel angewandt, welche ebenfalls häufig einer domänenspezifischen Struktur unterliegen.[1] Die Besonderheit der Zone-Analysis gegenüber der

[1] Diese Struktur wird auch IMRAD-Struktur bezeichnet, was für die Kategorien **I**ntroductions, **M**ethods, **R**esults **a**nd **D**iscussions steht. (Vgl. hierzu auch Sollaci & Pereira 2004)

Klassifikation vollständiger Dokumente liegt vor allem in der Kürze der Klassifikationstexte, welche meist nur einzelne Sätze oder Paragraphen umfassen.

1.3 Ziele und Umsetzung

Ein Ziel dieser Arbeit ist es, das bestehende Klassifikationsframework JASC durch eine weitere Klassifikationskomponente, die Support Vector Machine, zu erweitern. Als Grundlage hierfür dient LIBSVM[2], eine open source Bibliothek für Support Vector Machines. Eine Herausforderung bei diesem Vorhaben ist die schnittstellengerechte Adaption des Algorithmus in das bereits bestehende Framework. Die Integration in das Framework anstelle einer gesonderten Benutzung der Bibliothek soll gewährleisten, dass die unterschiedlichen Klassifikationsmethoden unter identischen Voraussetzungen und mit den im Framework integrierten Evaluationsverfahren verglichen werden können. Darüber hinaus kann die Komponente zur Merkmalsgenerierung und unterschiedlichen Erstellung von Dokumentvektoren genutzt werden. Dies ermöglicht nicht nur einen konkreten Vergleich von Support Vector Machines mit anderen gängigen Klassifikationsverfahren im Bereich der Zone-Analysis, sondern auch Rückschlüsse darüber, ob und welche Verfahren zur Vektorerstellung Einfluss auf die Klassifikationsergebnisse von Support Vector Machines haben.

Kapitel zwei dieser Arbeit befasst sich mit den Grundlagen des maschinellen Lernens und erläutert die grundlegende Arbeitsweise von ML-basierten Klassifikationssystemen. Hieraus ergeben sich die für die Klassifikation von Textdokumenten notwendigen Arbeitsschritte zur Präprozessierung und Repräsentation der Textdaten. Es werden gängige Verfahren zur Merkmalsauswahl und –gewichtung von Texten erläutert, die - passend zum Eingabeformat von Klassifikationsalgorithmen - zu einer Vektorrepräsentation von Textdokumenten führen. In Kapitel 3 werden Support Vector Machines als ein Verfahren zur ML-basierten Klassifikation vorgestellt. Kapitel 4 beschreibt die Einbindung dieses Klassifikationsverfahren in das JASC-Framework, auf dessen Grundlage schließlich Experimente zum Vergleich von Support Vector Machines mit anderen gängigen Klassifikationsverfahren vorgenommen werden.

[2] http://www.csie.ntu.edu.tw/~cjlin/libsvm/, zuletzt aufgerufen: 24.04.2015

2 Textklassifikation als maschinelles Lernproblem

Ein ML-basiertes Klassifikationssystem besteht aus einer Komponente zum Wissenserwerb und einer Komponente zur eigentlichen Klassifikation (vgl. Brückner 2004 S. 496). Der Wissenserwerb erfolgt durch die Analyse von vorklassifizierten Trainingsbeispielen. Aus diesen Beispielen werden Klassenprofile, sogenannte Modelle, erzeugt, die in der Klassifikationsphase als Entscheidungsgrundlage dienen. Ein Klassenmodell besteht üblicherweise aus einer Reihe von Merkmalen und dazugehörigen Gewichtungen, die aus den Trainingsobjekten dieser Klasse abgeleitet werden. Ein neues Objekt wird dann der Klasse zugeordnet, dessen Merkmalsprofil am besten zu den Merkmalsausprägungen des neuen Objektes passt. Diese Vorgehensweise impliziert eine Repräsentation der Klassifikationsobjekte als n-Tupel von Attribut-Wert-Paaren. Genauer werden Objekte als Vektoren repräsentiert. Identifiziert man jedes Merkmal/Attribut mit einer Vektordimension und den entsprechenden Vektoreintrag mit dessen Wert/Gewichtung, ist diese Darstellung äquivalent.

Im Falle von Textdokumenten als Klassifikationsobjekte setzt dieser Ansatz ein repräsentatives Trainingskorpus aus vorklassifizierten Dokumenten voraus. Ferner müssen auch diese als Vektoren repräsentiert werden, um als Eingabeformat eines Klassifikationsalgorithmus genutzt werden zu können

Dokumentvektoren

Entscheidend für den Erfolg eines Textklassifikationssystems ist nicht nur der Klassifikationsalgorithmus selbst, sondern auch eine angemessene Repräsentation der zu klassifizierenden Texte. Hierzu gehören die Auswahl repräsentativer Merkmale (bzw. der Ausschluss nicht-repräsentativer Merkmale), sowie eine zielorientierte Gewichtung der Merkmale.

Im Bereich der Textklassifikation bilden üblicherweise die Wörter eines Textes die potentiellen Merkmale einer Klassifikation. Es können aber auch n-Gramme über Wörter oder Buchstaben, sowie Wort-Paare oder ganze Wortketten verwendet werden. Im Falle von Wort-Merkmalen sind verschiedene Verfahren zum Ausschluss irrelevanter Merkmale üblich. So können beispielsweise inhaltslose Funktionswörter durch sogenannte Stoppwortlisten herausgefiltert werden. Gängig ist auch die Durchführung eines Stemmings, also die Zurückführung aller Wörter auf ihren Wortstamm. Solche Verfahren zur Merkmalsreduktion haben nicht nur das Ziel, überflüssige und eventuell undienliche Merkmale auszuschließen, sondern verkleinern auch die Dimension der Vektoren und somit die Laufzeit eines Klassifikationsalgorithmus.

Zur Gewichtung der ausgewählten Merkmale sind verschiedene Methoden geläufig. Zu den einfachsten Möglichkeiten Merkmalsvektoren zu generieren, gehören Verfahren, die auf den relativen Worthäufigkeiten im Korpus beruhen. Häufig wird das *Term-Frequency/Inverse-Document-Frequency*-Maß (*tf-idf*-Maß) verwendet. Hierbei wird die Häufigkeit eines Merkmals im Dokument mit der Anzahl der Dokumente, die dieses Merkmal ebenfalls enthalten, kombiniert. (Vgl. hierzu auch Heyer et al. 2008 Kap. 5) Eine weitere Möglichkeit ist die Verwendung des Loglikelihood Quotienten. Hierbei wird die Häufigkeit eines Merkmals im Dokument mit seiner Gesamthäufigkeit im Korpus verglichen. (Vgl. hierzu auch Zampieri et al. S. 31f.)

Obwohl zahlreiche Studien existieren, in denen verschiedene Verfahren zur Konstruktion von Featureräumen getestet werden (vgl. z.B. Siersdorfer & Sizov 2003), lässt sich die Frage nach einem geeignetsten Verfahren zur Bildung von Dokumentvektoren nicht universell beantworten. Dies liegt zum einen daran, dass unterschiedliche Klassifikationsalgorithmen durch unterschiedliche Vektorformate begünstigt werden. Vor allem ist die Textklassifikation aber ein Problem, das stark von den Texten selbst und der jeweiligen textspezifischen Domäne abhängt. So eignet sich ein Verfahren, dass sich für die Klassifikation vollständiger Dokumente bewährt hat, nicht zwingend auch für die Klassifikation sehr kurzer Textabschnitte oder Dokumente anderer Textsorten.

3 Support Vector Machines

Eine Support Vector Machine (deutsch: Stützvektormaschine) ist ein linearer Klassifikator, der auf dem Prinzip des maschinellen Lernens basiert. In ihrer Grundform sind Support Vector Machines binäre Klassifikatoren, das heißt, sie unterscheiden nur zwischen zwei möglichen Klassen. Die Ausgangsbasis von Support Vector Machines bildet, wie bei allen ML-basierten Klassifikatoren, eine Menge vorklassifizierter Trainingsobjekte. Diese werden als Vektoren in einem mehrdimensionalen Vektorraum aufgefasst. Jeder Vektor wird außerdem mit dem Label 1 oder -1 versehen, welches die Zugehörigkeit zu einer der beiden Klassen ausdrückt. Lineare Klassifikatoren versuchen eine gerade Trennlinie (im zweidimensionalen Raum) bzw. Hyperebene (im mehrdimensionalen Raum) in diesem Vektorraum zu finden, die die Objekte verschiedener Klassen voneinander trennt. Ist dies möglich, spricht man auch von linear separierbaren Daten.

3.1 Linear separierbare Daten

Ist ein Datensatz linear trennbar, gibt es in der Regel unzählige bis unendlich viele Möglichkeiten eine linear trennende Hyperebene zu finden. Support Vector Machines zeichnen sich dadurch aus, dass sie diejenige berechnen, dessen Abstand zu den ihr am nächsten gelegenen Trainingsobjekten maximal ist. Das Ziel ist also eine möglichst breite Grenzzone zwischen den Objekten verschiedener Klassen zu bilden. Die Objektvektoren, die den geringsten Abstand zu dieser Hyperebene haben, werden *Support Vectors* (Stützvektoren) genannt. Der Abstand der Stützvektoren zur Hyperebene wird als *Margin* bezeichnet. Da Support Vector Machines diesen Abstand maximieren, werden sie auch als large-margin-Classifier bezeichnet. Abbildung 3.1 zeigt eine Hyperebene mit Stützvektoren und Margin in einem zweidimensionalen Datensatz.

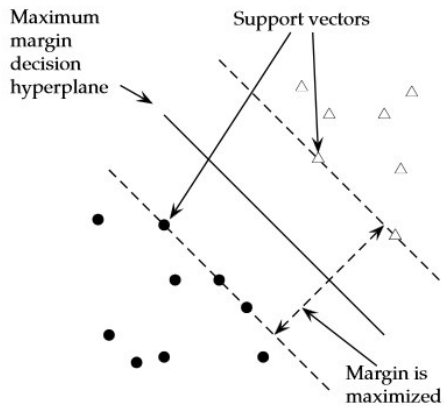

Abbildung 3.1: Linear trennende Hyperebene mit maximalem Margin und Support Vectors (Quelle: Manning et al. 2008, S. 294)

Die Trainingsphase von Support Vector Machines besteht aus der Berechnung dieser Hyperebene mit maximalem Margin. In der Klassifikationsphase wird diese als Entscheidungsfunktion genutzt. Je nachdem, auf welcher Seite der Hyperebene sich ein neues Objekt befindet, erfolgt die Zuweisung des Klassenlabels

Berechnung der Hyperebene

Mathematisch lässt sich eine Hyperebene in einem Vektorraum durch einen Richtungsvektor *w* und einen sogenannten bias *b* darstellen. Die Hyperebene bilden dann alle Punkte *x*, die die Gleichung

$$< w, x > + b = 0$$

erfüllen. $< , >$ bezeichnet hierbei das Skalarprodukt zweier Vektoren. Der Richtungsvektor *w* steht orthogonal (senkrecht) auf der Hyperebene und *b* gibt ihren Versatz zum Koordinatenursprung an. Durch diese Werte ist eine Hyperebene eindeutig bestimmt. Wie bereits erwähnt, muss die gesuchte Hyperebene zwei Bedingungen erfüllen: Sie muss den Trainingsdatensatz linear trennen und zusätzlich einen maximalen Margin erzielen. Unter allen linear trennenden Hyperebenen determiniert die Anforderung an maximalen Margin eindeutig die gesuchte Gleichung.

Eine Hyperebene, die einen Datensatz linear trennt, hat eine Eigenschaft, die sich ein linearer Klassifikator zu Nutze macht: Für alle Objektvektoren x_i, die sich im Vektorraum auf der der Hyperebene abgeneigten Seite befinden, ergibt die Ebenen-Gleichung $< w, x > + b$ einen negativen Wert. Für Vektoren auf der anderen Seite der Hyperebene wird der Wert positiv. Die

erste Bedingung, die die gesuchte Hyperebene erfüllen muss, lässt sich demnach wie folgt formulieren:

$$sgn(< x_i , w > +b) = label(x_i)$$

Für alle x_i im Trainingsdatensatz

Das Vorzeichen der Gleichung wird hierbei mit dem jeweiligen Klassenlabel des Trainingsvektors gleichgesetzt. Jede Hyperebene, die diese Eigenschaft erfüllt, trennt den Datensatz linear. Die Bedingung des maximalen Margins lässt sich mathematisch mit der Anforderung an minimale quadratische Norm des Gewichtsvektors *w* formulieren. Unter allen linear trennenden Hypereben, hat diejenige mit dem Richtungsvektor von kleinster quadratischer Norm

$$\| w \|_2 = \sqrt{\sum_i x_i^2}$$

den größten Abstand zu ihren Stützvektoren. Fasst man diese Bedingungen zusammen, lässt sich die Trainingsphase von Support Vector Machines als Optimierungsproblem wie folgt formulieren:

minimiere bezüglich:

$$\| w \|_2$$

so dass die Nebenbedingungen:

$$sgn(< x_i , w > +b) = label(x_i)$$

für alle xi des Trainingsdatensatz erfüllt sind

Ist die gesuchte Hyperebene gefunden, das heißt *w* und *b* berechnet, so dient diese als Entscheidungsfunktion zur Klassifikation für neue Objekte. Wie bei den Trainingsvektoren gibt das Vorzeichen der Ebenen-Gleichung ihre Position zur Hyperebene und somit ihre Klassenzugehörigkeit an:

$$class(x) = sgn(< w, x > +b)$$

3.2 Nicht linear separierbare Daten

Nicht jeder Datensatz ist linear separierbar. Dies kann an Messfehlern liegen, aber auch daran, dass die Verteilungen zweier Klassen überlappen. In diesem Fall ist es nicht möglich, eine gerade Grenze zwischen den Vektoren zweier Klassen zu ziehen. Insbesondere bei niedrigen Vektorraumdimensionen sinkt die Wahrscheinlichkeit linearer Separierbarkeit. Support Vector

Machines haben zwei Methoden mit dieser Art von Daten umzugehen, die in den beiden folgenden Abschnitten erläutert werden.

Einführung von Schlupfvariablen

Eine Möglichkeit im Umgang mit nicht linear trennbaren Trainingsdaten ist die Einführung sogenannter Schlupfvariablen. Diese Methode ermöglicht es, dem Algorithmus Fehler beim Einsetzen der Hyperebene zu erlauben. So dürfen einige Trainingsbeispiel innerhalb, oder auch auf der falschen Seite der breiten Margin-Zone liegen. Für jeden falsch positionierten Vektor werden dann Strafkosten berechnet, dessen Höhe relativ zum Abstand der Margin-Grenze ist. Zu diesem Zweck wird für jeden Trainingsvektor xi eine Schlupfvariable ξ_i eingeführt, die genau diesen fehlerhaften Abstand angibt. Ist also $\xi_i > 0$ verletzt xi die Nebenbedingung der Hyperebene. Abbildung 3.2 zeigt einen nicht-linear separierbaren Datensatz mit positiven Schlupfvariablen.

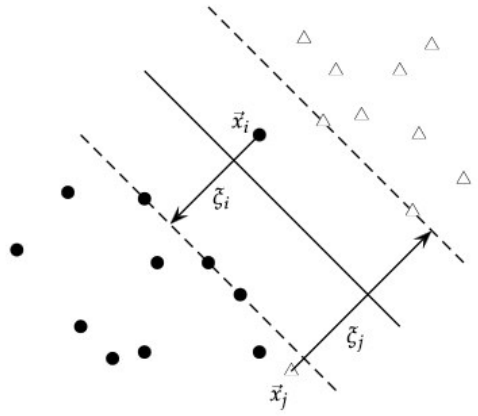

Abbildung 3.2: Nicht-linear trennbarer Datensatz mit Hyperebene und Schlupfvariablem. (Quelle: Manning et al. 2008, S. 300)

Das ursprüngliche Optimierungsproblem wird um die zusätzlichen Schlupfvariablen für jeden Vektor ergänzt. Der Algorithmus muss nun das beste Verhältnis finden zwischen einem möglichst großen Margin bei gleichzeitig möglichst geringer Summe aller Schlupfvariablen.

Der Kernel-Trick

Support Vector Machines haben eine weitere Methode mit nicht-linearen Daten umzugehen. Sie kommt dann zum Einsatz, wenn die lineare Separierbarkeit durch mehr als ein paar kleine, durch Schlupfvariablen ausgleichbare, Fehler, gestört wird. Um auch bei solchen Daten Support Vector Machines anwenden zu können, werden die Datenpunkte in einen höherdimensionalen Raum

transformiert. Die Grundidee hinter dieser Vorgehensweise fußt auf dem Theorem von Cover (vgl. Cover 1965). Das Theorem besagt, dass nicht-linear trennbare Daten durch die Transformation in einen ausreichend höheren Vektorraum in einen linear trennbaren Datensatz überführt werden können. In diesem Vektorraum kann folglich auch die entsprechende Hyperebene berechnet werden, die als Klassifikator für neue (ebenfalls transformierte) Objekte dient. Abbildung 3.3 zeigt einen nicht-linear trennbaren Datensatz im eindimensionalen Raum. Durch eine Überführung in den zweidimensionalen Raum wird der Datensatz linear separierbar und die Trennung durch eine Gerade wird möglich.

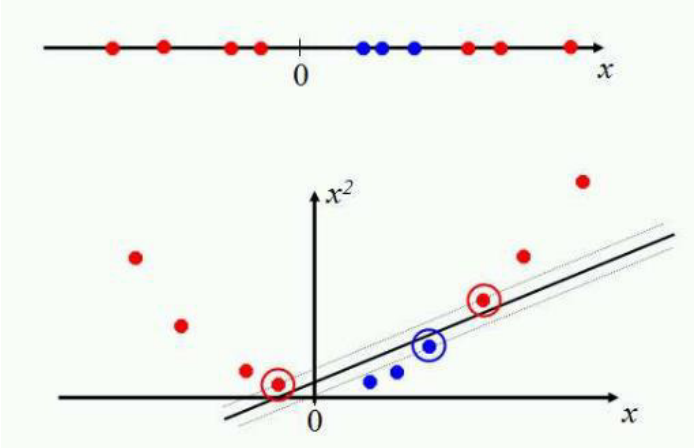

Abbildung 3.3: Ein nicht-linear trennbarerer Datensatz wird linear trennbar durch die Überführung in einen höherdimensionalen Raum. (Quelle: Manning et al. 2008, S. 304)

Zur Überführung eines Datensatzes in einen anderen Vektorraum muss in der Regel jeder Datenpunkt durch eine Transformationsfunktion f: x → f(x) neu berechnet werden, was einen hohen Berechnungsaufwand zur Folge hat. In das zu lösende Optimierungsproblem gehen die einzelnen Punkte jedoch nur als Skalarprodukt ein. Sofern es also eine Funktion K gibt, die das Skalarprodukt im neuen Vektorraum auf Basis der ursprünglichen Vektoren aus dem alten Vektorraum berechnen kann, braucht man die einzelnen Transformationen f(x) also gar nicht explizit auszurechnen. Es genügt eine Funktion K anzugeben, die alle nötigen Informationen liefert. Derartige Funktionen werden Kernel-Funktionen genannt. Für die Methode, das Skalarprodukt direkt und ohne Umwege über eine Transformationsfunktion zu berechnen, hat sich in der Literatur der Ausdruck Kernel-Trick eingebürgert.

.

4 Experiment: Zone-Analysis auf Stellenanzeigen

4.1 Aufbau des Experiments

Die Ausgangsbasis für das Experiment bilden etwa 300 vom BIBB zur Verfügung gestellte Stellenanzeigen. Anders als bei der herkömmlichen Textklassifikation werden bei einer Zone-Analysis Textabschnitte anstelle von vollständigen Dokumenten klassifiziert. Die zu klassifizierenden Einheiten sind somit nicht im Vorhinein gegeben, sondern müssen in einem vorgeschalteten Arbeitsschritt definiert und voneinander separiert werden. Im Klassifikation-Framework JASC übernimmt diese Aufgabe der *ParagraphUnitSplitter*. Die Analyse einiger Stellenanzeigen ergab, dass einzelne Sätze als Klassifikationsobjekte zu klein sind, da sich die einer Klasse zugehörigen Informationen meist über mehrere Sätze erstrecken. Mittels regulärer Ausdrücke und unter Berücksichtigung der typischen Formatierung von Stellenanzeigen, nimmt der *ParagraphUnitSplitter* daher eine Aufteilung der Stellenanzeigen in einzelne Paragraphen vor. Aus den ca. 300 Stellenanzeigen konnten dadurch etwa 1000 Paragraphen generiert werden. Nach einer manuellen Zuordnung in die zuvor definierten Klassen, bilden diese das Trainingskorpus für das Experiment. Der diesen Arbeitsschritten entsprechende Workflow wird in Abbildung 4.1 dargestellt.

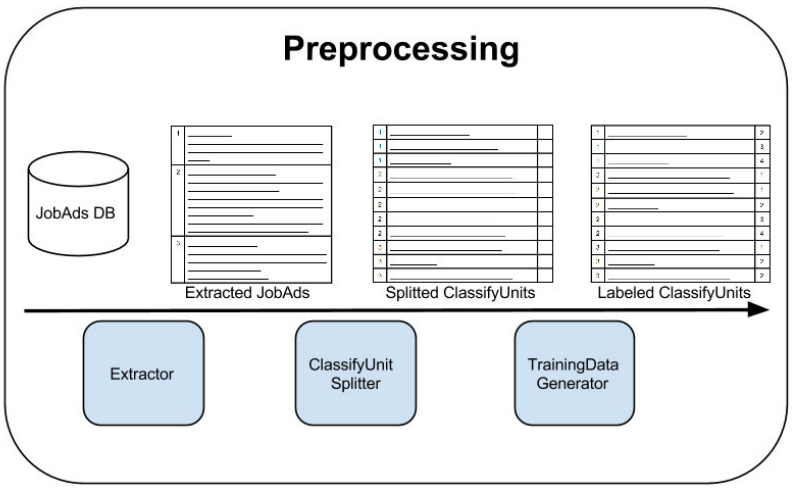

Abbildung 4.1: Preprocessing Workflow. (Quelle: Hermes 2015, S. 6)

Feature Engineering

Das Klassifikationsframework JASC stellt eine Reihe unterschiedlicher Verfahren zur Merkmalsauswahl, -Reduktion und Gewichtung bereit, die auf unterschiedliche Weise miteinander kombiniert werden können. Dies ermöglicht es, tausende unterschiedliche Verfahren

zur Erstellung von Dokumentvektoren zu vergleichen. Um den besten Kandidaten für die Klassifikation mit Support Vector Machines zu ermitteln, werden mehrere tausend Kombinationen aus Merkmalsauswahl, -Reduktion und –Gewichtung in einem Workflow (schematisch dargestellt in Abbildung 4.2) getestet.

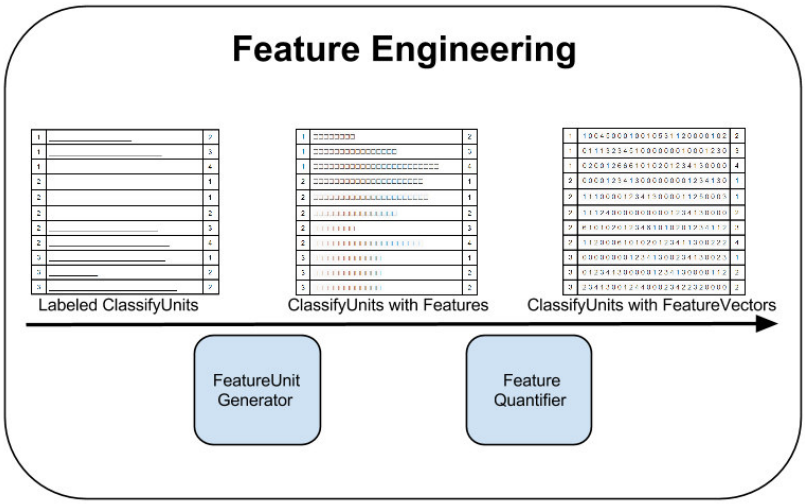

Abbildung 4.2: Feature Engineering Workflow. (Quelle: Hermes 2015, S. 7)

Klassifikation und Evaluation

Die unterschiedlichen Modellbildungsverfahren werden anschließenden in einzelnen Klassifikationsvorgängen getestet. Dazu wurden die vorklassifizierten Paragraphen in zehn Gruppen unterteilt, von denen in einem Kreuzvalidierungsverfahren jeweils neun als Trainingsdaten und eine als Testdaten verwendet wurden. Über den Vergleich mit dem Gold-Standard der manuell vorgenommenen Vorauszeichnungen werden die gängigen Evaluationswerte Precision, Recall, Accuracy und F-Score ermittelt.

Neben der Klassifikation mit Support Vector Machines werden auch die bereits zuvor im Framework integrierten Klassifikationsverfahren getestet. Hierzu zählen der Rocchio-Algorithmus, ein ebenfalls lineares Klassifikationsverfahren, ein Naive Bayes Klassifikator und ein k-Nearest-Neighbour Klassifikator (zu den einzelnen Klassifikationsverfahren siehe auch Manning et al. 2008). Außerdem wurde ein auf regulären Ausdrücken basierender, regelbasierter Klassifikator implementiert, der, den anderen Verfahren vorgeschaltet, die Klassifikationsergebnisse signifikant verbesserte (Vgl. Hermes 2014 S. 7) Zur Evaluation der Support Vector Machines werden sämtliche Klassifikationsdurchläufe sowohl mit als auch ohne regelbasierte Vorklassifikation getestet.

4.2 Ergebnisse

Zum direkten Vergleich der unterschiedlichen Klassifikationsverfahren wurden zunächst alle Klassifikationsdurchläufe ohne den vorgeschalteten regelbasierten Klassifikator ausgeführt. Die jeweils besten Evaluationsergebnisse aller Klassifikationsverfahren, sortiert nach den besten F-Score-Werten, sind in Tabelle 4.1 aufgeführt.

f-score	Klassifikator	Distanzmaß	Merkmale	Merkmals-reduktion	Merkmals-gewichtung
0,994	SVM	-	2Gramme 3Gramme	sw-filter	loglikelihood
			3Gramme	sw-filter	
			2Gramme 3Gramme	sw-filter stemming	
			3Gramme	sw-filter stemming	
			3Gramme	sw-filter normalize stemming	
			2Gramme 3Gramme	sw-filter normalize stemming	
0,993	SVM	-	3Gramme	sw-filter normalize	loglikelihood
0,992	SVM	-	2Gramme 3Gramme	sw-filter normalize	loglikelihood
			3Gramme	sw-filter	TF-IDF
0,992	KNN(k=1)	Cosinus	3Gramme	sw-filter normalize stemming	loglikelihood
			Wörter	-	TFIDF
0,9610	KNN(k=4)	Cosinus	2Gramme 3Gramme	-	loglikelihood
0,9086	Rocchio	Cosinus	3Gramme	-	loglikelihood
0,7689	NaiveBayes	-	Wörter	sw-filter stemming	-

Tabelle 4.1: Klassifikationsergebnisse ohne vorgeschaltete regelbasierte Klassifikation

Von allen Klassifikationsverfahren liefern Support Vector Machines die besten Ergebnisse. Hierbei fällt insbesondere auf, dass unterschiedliche Verfahren zur Merkmalsgenerierung offenbar kaum bis keinen Einfluss auf das Klassifikationsergebnis haben. So bleibt der F-Score bei unterschiedlichen Kombinationen zur Merkmalsselektion nahezu identisch. Die Verwendung von Wörtern verschlechtert das Ergebnis ebenfalls nur um weniger als 1%. (Der beste F-Score mit Wortmerkmalen liegt bei 0,9915). Genauso gering ist der Unterschied zwischen dem tf-idf-Maß und loglikelihood-Quotienten zur Gewichtung der Merkmale. Beinahe genauso gut schneidet der KNN-Klassifikator ab. Hier ist allerdings zu beobachten, dass die Ergebnisse nur bei der Berücksichtigung eines einzigen Nachbars (k=1) mit den Ergebnissen von Support Vector Machines mithalten können. Offenbar gibt es zu jedem Paragraphen im Testset einen – hinsichtlich der gewählten Merkmale – ähnlichsten Paragraphen im Trainingskorpus, der mit dem richtigen Klassenlabel versehen ist. Dies sorgt innerhalb des begrenzten Testkorpus zwar für sehr

gute Ergebnisse, zu bedenken ist jedoch, dass bei einer Vergrößerung des Korpus, das Heranziehen nur eines Nachbarn möglicherweise nicht mehr ausreichen könnte (vgl. Hermes 2015). Aus diesem Grund wurden auch Messungen unter Berücksichtigung mehrerer Nachbarn durchgeführt, die, wie in der Tabelle abzulesen, zu etwas schlechteren Klassifikationsergebnissen führten. Interessant ist auch der Vergleich von Support Vector Machines mit dem ebenfalls linearen Rocchio-Klassifikator. Dieser schneidet mit einem F-Score von 0,9086 deutlich schlechter ab. Noch niedriger ist der F-Score des besten Naive-Bayes Verfahren, welcher mit 0,7689 weit zurückfällt.

Ähnliche Beobachtungen lassen sich bei der Klassifikation mit vorgeschalteter regelbasierter Klassifikation machen (Abbildung 4.2). Die Ergebnisse der Support Vector Machines und des KNN-Klassifikators (mit k=1) bleiben nahezu identisch. Dies liegt daran, dass beide Verfahren bereits allein eine sehr hohe Precision (0,996) haben, die bereits alle, durch reguläre Ausdrücke klassifizierbaren Paragraphen, beinhaltet. Alle anderen Verfahren, sowie die KNN-Klassifikation mit mehr als einem Nachbarn können durch die regelbasierte Klassifikation ihre Precision (und somit auch den F-Score) leicht verbessern.

f-score	Klassifikator	Distanzmaß	Merkmale	Merkmals-reduktion	Merkmals-gewichtung
0,994	SVM	-	2Gramme, 3Gramme	sw-filter	llh
0,993	SVM	-	3Gramme,	sw-filter, stemming normalize	loglikelihood
			2Gramme 3Gramme	sw-filter stemming normalize	
			3Gramme 2Gramme	sw-filter stemming	
			3Gramme	sw-filter	
			3Gramme	sw-filter stemming	
0,992	SVM	-	2Gramme 3Gramme	sw-filter, normalize	loglikelihood
			3Gramme	sw-filter, normalize	
0,9915	SVM	-	3Gramme	sw-filter	TF-IDF
			2Gramme 3Gramme	sw-filter stemming	
			3Gramme	sw-filter stemming	
0,991	KNN (k=1)	Cosinus	2Gramme	sw-filter, stemming, normalize	loglikelihood
			3Gramme	sw-filter, stemming, normalize	
			3Gramme	sw-filter	
0,9676	KNN (k=4)	Cosinus	2Gramme, 3Gramme	sw-filter, stemming	loglikelihood
0.9222	Rocchio	Euklid	Wörter	sw-filter, stemming	loglikelihood
0,7689	NaiveBayes	-	Wörter	sw-filter stemming	-

Tabelle 4.2: Klassifikationsergebnisse mit vorgeschalteter regelbasierter Klassifikation

5 Fazit

Durch die Integration von Support Vector Machines in das Klassifikationsframework JASC konnten verschiedene Experimente zum direkten Vergleich dieses Klassifikationsverfahren mit anderen gängigen Klassifikationsalgorithmen durchgeführt werden. Insgesamt wurden über 2000 verschiedene Experimente ausgeführt, bei denen jeweils unterschiedliche Kombinationen zur Merkmalsgenerierung und –gewichtung getestet wurden. Die Experimente haben gezeigt, dass mit Support Vector Machines die besten Klassifikationsergebnisse erzielt werden können. Als ein großer Vorteil bei der Klassifikation mit Support Vector Machines hat sich ihre Robustheit gegenüber unterschiedlichen Methoden zur Merkmalsauswahl und -selektion erwiesen. Experimente ohne Merkmalsselektion und auf der Basis von Wortmerkmalen erzielten einen ähnlich hohen F-Score (0,9915) wie Experimente auf N-Grammen, bei denen alle Selektionsverfahren angewendet wurden (0,994). Dieser Vorteil lässt sich nicht nur anhand der hier durchgeführten Experimente beobachten, sondern auch mathematisch erklären: Einfluss auf die Positionierung der linear trennenden Hyperebene – und somit auch auf die Entscheidungsfunktion zur Klassifikation neuer Objekte – haben nur die Stützvektoren des Trainingssets, also die Vektoren, die der Hyperebene am nächsten liegen und den Margin determinieren. Die Lage aller anderen und somit der meisten Vektoren hat dahingegen keinen Einfluss auf die Hyperebene. Solange ihr Abstand zur Hyperebene nicht unter dem der Stützvektoren liegt, sind sie beliebig im Vektorraum positionierbar. Der Algorithmus analysiert damit eigenständig, welche Merkmale zur Klassenbildung relevant sind, da nur die Stützvektoren und somit auch nur die in den Stützvektoren vertretenen Merkmale die Klassifikation beeinflussen. Die im Rahmen dieser Arbeit durchgeführten Experimente auf Stellenanzeigen haben dies bestätigt. Zusammenfassend lässt sich also festhalten, dass durch die Klassifikation mit Support Vector Machines die Notwendigkeit einer aufwändigen Merkmalsanalyse entfällt.

6 Literaturverzeichnis

COVER, T. M. (1965) *Geometrical and Statistical properties of systems of linear inequalities with applications in pattern recognition.* IEEE Transactions on Electronic Computers. EC-14: 326–334.

BRÜCKNER, T. (2004) *Textklassifikation.* In: Carstensen et al. (eds) (2004) Computerlinguistik und Sprachtechnologie. Eine Einführung. Elsevier, München.

HERMES, J. (2015) *Text Mining auf Stellenanzeigen.* Erscheint in: Wissenschaftliche Diskussionspapiere, Bundesinstitut für Berufsbildung.

HEYER, G. et al. (2008) *Text Mining: Wissensrohstoff Text. Konzepte, Algorithmen, Ergebnisse.* W3L, Herdecke, Bochum.

MANNING, C. D. & RAGHAVAN, P. & SCHÜTZE, H. (2008) *Introduction to Information Retrieval.* Cambridge University Press, New York.

SEBASTIANI, F. (2002) *Machine learning in automated text categorization.* ACM Computing Surveys (CSUR), 34(1), 1-47.

SIERSDORFER, S. & Sizov, S. (2003) *Konstruktion von Featureräumen und Metaverfahren zur Klassifikation von Webdokumenten.* Datenbanksysteme für Business, Technologie und Web, Tagungsband der 10. BTW-Konferenz. 197-206.

SOLACCI, L. B. & PEREIRA, M. G. (2004) *The introduction, methods, results ans discussion (IMRAD) structure: a fifty-year survey.* Journal of the Medical Library Association (JMLA), 92(3), 364-7.

WAPNIK, W. (1995) *The Nature of Statistical Learning Theory.* Springer, New York.

ZAMPIERI, M. & HERMES, J. & SCHWIEBERT, S. (2013) *Identification of Patterns and Document Ranking of Internet Texts: A Frequency -based Approach.* In: Zampieri, M. & Diwersy, S. (eds.) (2013) Non-standard Data Sources in Corpus-based Research. Shaker, Aachen.