Contents

1 Einführung

Die folgende Arbeit befasst sich mit der Frage inwiefern sich Feature Engineering bei der Anwendung verschiedener Klassifizierungsmethoden auf die Ergebnisse auswirkt. Der ganze Versuch wird in Zusammenarbeit mit der Firma Osram Opto Semiconductor unternommen.

1.1 Motivation

Digitalisierung ist heutzutage eines der meistverwendeten Buzzwords der modernen Industrie. Arbeitsschritte sollen automatisiert oder generell "smarter" gestaltet werden. So auch in der Sales-Abteilung bei Osram OS. Ein großer Bestandteil der Arbeit eines Mitarbeiters im Sales-Department ist die Kundenbetreuung, welche fast ausschließlich über E-Mailverkehr vollzogen wird. Diese Arbeit schließt auch das bearbeiten und einordnen der jeweiligen Anfragen ein. Welcher Kunde oder interessierter Kunde will welche Informationen oder Auskünfte von mir?
Um dem mühsamen und oft zeitaufwendigen Prozess zu vereinfachen, soll getestet werden wie sehr dieser Sortierungsvorgang maschinell durchgeführt werden kann. Der Baustein der Klassifizierung ist nur einer der Schritte zur vollautomatisierten Beantwortung oder Bearbeitung der Anfragen. Solche Anfragen können Quote (ungefähre Preis für ein Bauteil), Pricing (Genauer Preis für Bauteile in Anbetracht der wirtschaftlichen Lage und des Kunden) oder allgemeine Kundenanfragen beinhalten. Das Ziel ist beispielsweise Bestellungen automatisiert zu bearbeiten ohne, dass ein Mitarbeiter eingreifen muss. Diese Arbeit betrachtet nur den Baustein der Klassifizierung der E-Mails.

1.2 Anforderungen

Der Forschungsinhalt der Arbeit legt den seinen Schwerpunkt auf das *Preprocessing* der E-Mails und das Verwenden verschiedener Klassifizierungsmethoden. Preprocessing bedeutet dass Datensätze (hier E-Mails) vorab angepasst oder verändert werden, damit leichter oder effektiver mit ihnen gearbeitet werden kann. Dafür muss der vorhandene Datensatz (E-Mail Anzahl) zum Teil per Hand durchsucht werden um festzustellen welche Mechanismen sinnvoll anzuwenden sind. Solche Mechanismen werden *Features* genannt.
Nachdem sinnvolle *Features* ausgewählt worden sind, werden verschiedene Klassifzierungsarten ausgewählt. Diese werden je nach Bedarf trainiert (Neuronale Netze, oder Support Vektor Machines, wird unten erklärt) oder einfach mit den bearbeiteten Datensätzen gefüttert. Nach dem Ablauf der Klassifizierung werden die einzelnen Ergebnisse verglichen.

1.3 Forschungsfrage

Das Ziel der Forschung liegt in der Frage welche angewandten Features ,mit dem vorhandenem Datensatz, die besten Ergebnisse liefert. Dabei werden nicht nur

die Features verglichen sondern auch die Klassifizierer. Welcher Klassifizierer ordnet mit der geringsten Fehlerrate zufällig ausgewählte Testmails ein? Außerdem liegt ein besonderes Augenmerk auf der Verwendung eines Neuronalen Netzes und wie gut oder schlecht es zum Vergleich der anderen Technologien ab.

2 Theoretischer Teil

Es gibt verschiedene Ansätze um eine Klassifizierung durchzuführen. In unserem Fall verwenden wir zum einen Support Vektor Machine, K-nearest Neighbour und Neuronale Netze. Alternativen die zur verfügung stehen werden auch knapp beleuchtet.

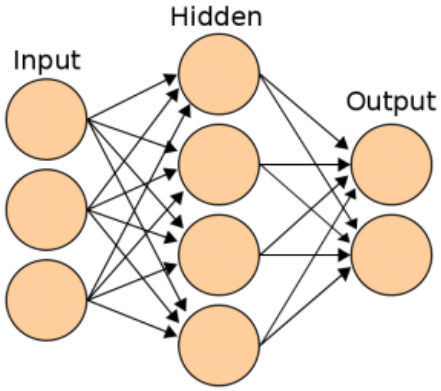
2.1 Grundlagen Künstlicher Neuronaler Netze

2.1.1 Analogie zum menschlichen Nervensystem

Um ein Künstliches Neuronales Netz anschaulich zu erklären wird gern ein Vergleich mit dem menschlichen Nervensystem gezogen.(Holger Schulze) Das Gehirn des Menschen enthält bis zu 100 Milliarden Nervenzellen, welche durch Verwendung oder nicht Verwendung unseren biologischen Lernprozess darstellen. Die Verarbeitung im Gehirn wird durch das bekannte SOR-Modell (Stimulus Organismus Response) dargestellt. Dieser Vorgang kann analog für ein künstliches Neuron, auch Unit genannt, angewendet werden. Dafür werden drei separate mathematischen Operationen ausgeführt(Bildung des Inputs, Bildung des Aktivitätsniveaus, Bildung des Outputs) . ABBILDUNG Ein KNN teilt sich in drei verscheidene Schichten auf.
-Input Layer: Der Input Layer nimmt alle eingehenden Umgebungsvariablen ungefiltert auf und leitet diese an den Hidden Layer weiter.

-Hidden Layer: Der Hidden Layer ist unsichtbar für außenstehende und dient zur Informationsverarbeitung und Verteilung. Es ist möglich mehrere Hidden Layer hintereinander zu legen. Dadurch entseht eine Tiefe, die die Komplexität des Netzwerks vergrößert, aber auch ihr Leistungsvermögen steigert.

- Output Layer: Der Output Layer verarbeitet die vorprozessierten Informationen weiter und gibt diese an ein konkretes Ziel weiter.



2.1.2 Deep Learning

https://jaai.de/machine-deep-learning-529/ Um eine KNN in vollem Umfang sinnvoll zu nutzen, muss das Netzwerk trainiert werden. Nachdem das Netzwerk aufgebaut wurde, erhält jedes Neuron eine zufällige Gewichtung, also einen Wert. Beim ersten Durchlauf werden die Eingangsdaten vom Input Layer an die Hidden Layer weitergegeben. Die Neuronen des Hidden Layer geben die Daten weiter an die folgenden Schichten und gewichten diese mit ihrem zufällig erhaltenem Wert. Die Werte ergeben ein Gesamtergebnis, welches durch den Output Layer evaluiert wird. Dieser Prozess nennt sich *Forward Pass*.
Dieses erste Ergebnis ist natürlich ein Zufallsprodukt, da die Gewichtung der Neuronen zufällig gewählt wurden. Um das KNN zu trainieren wird oft das *Supervised Learning* verwendet. Damit ist das Lernen anhand von Beispielen gemeint. Das läuft Folgendermaßen ab:
Das Netz wird mit einer festen Menge an Inputdaten gefüttert. Das Ergebnis wird mit einem passenden Beispieldatensatz verglichen. Danach werden positive und negative Abweichungen durch beispielsweise den Mittelwert der Quadratischen Abweichung errechnet.
Mit dem *Backwardpass* werden nun die Neuronen angepasst. Der entstandene Fehler durch den *Forwarpass* wird rückwirkend Schritt für Schritt durch das Netz nachgestellt. An jedem Neuron wird festgestellt wie hoch der Einfluss der Gewichtung auf das Endergebnis war und dementsprechend die Gewichtung verändert, sodass der Fehler ein wenig näher an das Wunschergebnis kommt. Ein wichtiger Wert für die Anpassung der Gewichtungen ist die sog. *Learning rate*. Dieser Wert Beschreibt in welchem Ausmaß eine Gewichtung in einem Lerndurchlauf verändert werden darf. Ist dieser Wert zu hoch gewählt kann das Minimum der Lernkurve verfehlt werden (Overshooting). Deshalb werden im späteren Lernverlauf die Werte der *Learning rate* immer kleiner.
Der ganze oben erklärte Ablauf von Forward Pass bis Backward Pass wird *Epoche* genannt. Wenn ein KNN trainiert wird, kann man festlegen wie viele Epochen geplant sind um das Netz zu verbessern. Die Annäherung an eine bestimmte Trefferrate nennt man Konvergenz. Falls das Trainieren die Trefferrate

nicht mehr signifikant verändert, kann der Trainingsprozess auch schon vor Abschluss aller Epochen beendet werden. Das erspart Zeit und Rechenleistung.
Beispiel:
Ein KNN wird auf das Erkennen von Bildern trainiert. Die Erkennungsrate ob auf einem Bild ein Frosch ist oder nicht liegt nach 25 Epochen bei 80,0003 %. Geplant sind 100 Epochen für den Trainingsdurchlauf. Nach 30 Epochen liegt der Wert bei 80,0004 % und somit liegt kaum eine Verbesserung vor. Hier kann der Trainingsprozess abgebrochen werden.

2.2 Datenvorbereitung

Eine KI egal welcher Art braucht Trainingsdatensätze um zu lernen. Es gibt einige Faustregeln für die Eigenschaften der Datensätze. Zum einen trifft hier die Regel "je mehr desto besser" vollkommen zu. Je mehr Datensätze eine KI zur Verfügung hat, desto genauer kann sich das KNN auf jede Art von Anfrage einstellen. Dadurch das bei einer großen Menge an Daten viele Sonderfälle mit enthalten sein könnten wären diese ebenfalls abgedeckt.
Ein weiterer wichtiger Faktor ist die Balance der Datensätze. Nehmen wir an eine KI muss zwischen relevanten E-Mails von Kunden und Spam oder unwichtigen E-Mails unterscheiden. Ihr Trainingsdatensatz besteht aber aus 80% relevanten Mails. Hier kann das sogenannte *class imbalance problem* auftreten. Bei eine fehlenden Balance der Dateien ist es für die KI schwer zu erkennen wann eine E-Mail eine Spam-Mail oder eine Kundenanfrage ist. Die Trefferrate leidet signifikant unter den Startbedingungen. Entweder müssen wesentlich mehr Trainingsiterationen durchgeführt werden oder der Lerndatensatz wird angepasst.
Ein weiteres Problem für machine learning generell ist *Overfitting* oder *Onderfitting*. Beim Overfitting wurde das machine learning System "zu genau" auf die Problemstellung trainiert. Das bedeutet Fehlerfälle oder oder Sonderfälle würden bei echten Testdaten falsch klassifiziert werden, da diese nicht trainiert worden sind und das System zu unflexibel geworden ist. Generalisierung ist beim Einsatz von KI-Systemen unabdingbar. Beim Underfitting ist einfach das Gegenteil zum Overfitting. Schlecht trainierte KI's liefern schlechte Ergebnisse.
Ein Sonderfall der Datenvorbereitung sind sogenannte *Outlier*. Ein Outlier sind beispielsweise falsch betitelte Trainingsdaten oder zu große sonderfälle. Beim Usecase der E-Mail-Klassifizierung wären das beispielsweise zehnfach längere E-Mails. Wie gut ein KI-System ist, zeigt sich mit dem Umgang solcher Outlier. Je robuster ein System desto weniger Einfluss hat es auf das Ergebnis.

2.3 Support Vector Machine

2.3.1 Funktionsweise

Eine sehr weit verbreitete Variante der Klassifizierung sind sogenannte *Support Vector Machines*. Diese Methode benötigt ,wie auch ein KNN, vorab aufbereitete Trainingsdaten. Jedes Trainingsobjekt wird als Vektor in einem Vektorraum

dargestellt. Die Aufgabe des Algorithmus ist es in den Vektorraum eine Trennfläche, eine Hyperebene, einzubauen. Dabei ist zu beachten, das es mehrere Hyperebenen geben kann. Es wird die Hyperebene gewählt, welche den maximalen Abstand zu den anliegenden Vektoren inne hat. Vektoren die nicht im Bereich der Hyperebene liegen nennt man Stützvektoren da sie gebraucht werden um die Ebene mathematisch exakt zu beschreiben. Eine Voraussetzung um diese Ebene zu errechnen ist, dass alle Objekte linear trennbar sind. Bei der Verwendung von realen Trainingsobjekten ist das in der Regel nicht möglich. Die Vektor Machine verwendet einen sog. Kernel-Trick um dieses Problem zu umgehen. Dieser beinhaltet das Überführen des Vektorraumes in eine höheren dimensionalen Raum, teilweise bis ins Unendliche. Dadurch können selbst die komplexesten Vektorräume linear getrennt werden. Die Nachteile der Anwendung liegen klar auf der Hand. zum einen muss viel Rechenaufwand betrieben werden um den Vektorraum zu transformieren und zum anderen ist die Darstellung der Hyperebene im niederen dimensionalen Raum zu ungenau, also nicht zu verwenden. Der Kernel-Trick bedient sich hier der Stützvektoren um das berechnen der Trennebene so zu gestalten, dass bei der Rücktransformation die Hyperebene trotzdem zu verwenden ist.

2.3.2 Vergleich zu Künstlichen neuronalen Netzen

Da SVM als eine etablierte Klassifizierungsmethode ist, sollte man den Vergleich zu Neuronalen Netzen begutachten. Eine Forschung der des Institut für Organische Chemie und Chemische Biologie, Johann Wolfgang Goethe-Universität Frankfurt verglich Künstliche Neuronale Netze und SVM bei einer binären Klassifizierung von Chemischen Substanzen. Für uns ist nur der Einsatz von zwei Topologien Neuronaler Netze und einer Support Vektor Machine interessant. Die zwei Topologien der KNN unterschieden sich nur bei zwei direkten Verbindungen zwischen Inputlayerknoten und dem einzigen Outputlayerknoten. Alle drei Varianten wurden mit den selben Datensätzen trainiert.
Es konnten mehrere Erkenntinsse aus dem Veruch gezogen werden. Zum einen belief sich die Präzision aller Technologien im Bereich von 80% , was sehr hoch ist.Die SVM Variante war nach gleichen Trainingszyklen trotzdem 1-2% präziser. Außerdem konnte beobachtet werden, dass beide KNN's, obwohl ihre Neuronenanzahl gleich groß war, unterschiedliche *Faultpredictions* lieferten.
Generell schneiden Support Vektor Machines bei binären Klassifizierungen minimal besser ab, nur ist das nur bei zwei verschiedenen recht einfachen Neuronalen Netzen getestet worden. Die Menge der Neuronen im Hidden Layer spielt bei der Präzision eine große Rolle.

2.4 k-nearest Neighbor

Damit es nicht zu Verwechslungen kommt bedeutet KNN in diesem Absatz k-nearest Neighbor, nicht wie oben Künstlich Neuronales Netz.
KNN ist eine der älteren Klassifizierungsmethoden und dazu noch eine sehr einfache. Das Grundprinzip des Algorithmus basiert auf *lazy learning* und

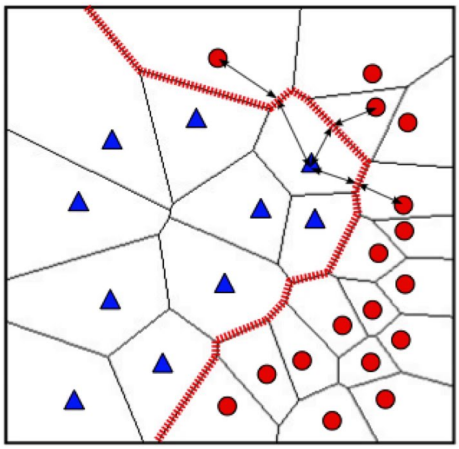


Figure 1: Beispiel einer Voronoi-tessalation

Abständen in Diagrammen. Lazy Learning bedeutet nichts anderes als das einfache Abspeichern der Trainingsdaten. Diese Trainingsdaten werden in Samples aufgeteilt und in einen Vektorraum umgewandelt. Die Abstände der Trainingssamples wird durch den Euklidischen Abstand bestimmt. Formel ??
Dadurch dass nur begrenzte Trainingssamples vorhanden sind, wird bei der Darstellung klar, das Bereiche um Samples nicht einfach freier Raum sind, sondern zur Klasse des Samples zählen. Ein solcher Raum wird *Voronoizelle* genannt. Dieser ergibt sich durch den Vergleich zu anderen Punkten im Diagramm. Dadurch dass der euklidische Abstand benutzt wird, ist eine Voronoizelle der Raum in dem ein möglicher Punkt immer am nächsten dieser einen Zelle im Vergleich zu anderen liegt. Eine Voronoizelle ist immer ein Polygon. Ein Beispiel für ein Voronoi-Diagramm ist in Grafik 2 dargestellt.
Diese Raumaufteilung erleichtert das Klassifizieren von Testsamples ungemein. Das Testobjekt wird zu der Klasse gezählt, welche der Klasse der Voronoizelle entspricht. Bei einer größeren oder engeren Ansammlung der Trainingsdaten, werden die Zellen automatisch kleiner und somit wird die Klassifizierung genauer. Die Vorteile liegen klar auf der Hand. Im Verglich zu SVM oder Entscheidungsbäumen ist eine viel flexiblere Unterteilung der Datensätze möglich. Die beiden anderen Ansätze verwenden Hyperebenen oder vertikale Unterteilungen. Der KNN Algorithmus kann sogar Parabeln oder Hyperbeln darstellen, was für so einen einfachen Algorithmus erstaunlich ist.
Leider bringt Flexibilität auch ihre Nachteile mit sich. Der Algorithmus overfittet sehr leicht und hat große Probleme mit Outliern. Beispielsweise wird ein Testsample falsch gelabled. Bei anderen Methoden würde dies mit der Zeit irrelevant werden und hat wenig Einfluss. Für KNN ist der Einfluss auf das Ergebnis enorm, da jedes Testsample, welches in der falsch ausgeschriebenen Zelle liegt, wird auch falsch gelabled. Das wird dadurch verstärkt, dass ein KNN-Algorithmus keinerlei Confidencelevel erstellt, somit gehen solche Fehler

komplett unter.
Um dem entgegenzuwirken und den Algorithmus stabiler zu machen, vergleicht man nicht nur ein Sample mit dem Testsample sondern mehrere gleichzeitig. Um ein Testsample zu einer Klasse zuzuordnen wird ein "k" bestimmt, welches festlegt wie viele Samples mit dem Testsample verglichen werden soll. Beispielsweise wird k=3 festgelegt, dann wird nach den drei samplen gesucht, die am nächsten an unserem Testsample liegen. Bei einer Zweiklassenaufteilung wird das Testsample zu der Klasse zugeordnet, welche die zwei von drei Samples stellt. Im Idealfall wird K immer ungerade gewählt und nicht als ein Vielfaches der vorhandenen Klassen. Das erspart falsche Klasseneinteilung die zustande kommen würde, wenn nur das nächste Sample verglichen werden würde.

2.5 Alternativen

2.5.1 Label Propagation based algorithm

In betracht der Umstände Ein indisches Forscherteam hat im Gebiet der Textklassifizierung eine neue Art der *weakly supervised text classification* angewandt. Weakly supervised bedeutet, dass für einen Lernprozess nur wenige leicht zugängliche aber ausschlaggebende *labeld data* verwendet wird. Labeld data wiederum ist als Anfangsdatensatz zu verstehen, welcher im Voraus von Menschen in gewisse Klassen sortiert oder mit aussagekräftigen Tags versehen worden ist. Beispielsweise ob eine bestimmte Anzahl eingehender E-Mails nur Bestellungen sind, dann sind alle Dateien mit dem Tag "Bestellung" gezeichnet. Der Ansatz ist die Technik der *Label Propagation* mit *Topic Modelling* zu kombinieren. Label Propagation ist die Repräsentation von labeled und unlabled Data als Konten in einem Graphen. Die Kanten des Graphen spiegeln die Gleichheit der Knoten wieder. Knoten geben ihre label-Information an ihre Nachbarn weiter da sie im Graphen im selben Bereich liegen. Dieser Prozess läuft so lange ab bis alle Knoten durch ihre Lage mit dem Passenden Label gekennzeichnet sind.
Das zweite Schlüsselelement der Herangehensweise ist das *Topic Modelling*. Diese Technik ist im Bereich der Natural Language Processing Sparte sehr verbreitet. Das Team benutzt *Latent Dirichlet Allocation* als schon vorhandenes Topic Model. LDA arbeitet sich durch alle vorhandenen Dokumente und erarbeitet durch das vergleichen der Auftrittshäufigkeit mehrerer Schlüsselwörter einen passenden Überbegriff, oder hier, ein Topic für jedes Dokument.
Die Textklassifizierung dieses Ansatzes hat mehrere, im Einsatz befindende, Technologien übertroffen. Und neue Wege zu einer schnelleren Verarbeitung aufgezeigt.

2.5.2 E-Mail Classification with Co-Training

Ein Alternativer Ansatz ist die Verwendung von Co-Training Algorithmen. Co-Training benutzt, als Lernprozess bei der Klassifizierung, ebenfalls eine kleine

Menge labeled-Data. Beim Beispiel der Klassifizierung werden zwei verschiedene *classifier* erstellt, welche sich jeweils um ein anderes Klassifizierungsproblem kümmert. Beide Elemente werden mit den wenig vorbereiteten Daten trainiert. Danach arbeiten beide nur mit unlabled-data und unterstützen sich gegenseitig. Wie genau funktioniert das?
Classifier1 bekommt einen unbearbeiteten Datensatz zur Bearbeitung, welcher mit einer hohen Wahrscheinlichkeit zu einer bestimmten Klasse zu zählen ist. Classifier1 labeled also das Ergebnis, welches danach auch für Classifier2 gelabeled ist. Für Classifier2 wäre die Entscheidung welche Klasse der Datensatz hat nicht so leicht gefallen, da es eine andere initiale Problemstellung erhalten hat. Somit arbeiten sich beide Classifier gegenseitig zu und unterstützen sich. Untersuchungen zu Folge kann der Einsatz von Co-Training die Präzision nach gleichen Trainingsiterationen um den Faktor 2 verbessern.
Weitere Erkenntnisse wurde durch das Anwenden der Methode mit Support Vector Machines (SVM) und Naive Bayes erlangt. Zum einen ist die Accuracy(ob das Klassifizieren korrekt war) stark von der Balance des Datensatzes abhängig. Außerdem ist die Effektivität der Methode sehr vom verwendeten Algorithmus abhängig. SVM erzielte deutlich bessere Ergebnisse als Naive Bayes, welches sogar negative Trainingsergebnisse bei schlecht balancierten Datensätzen lieferte.

2.6 Feature Selection Metrics

2.7 Text Pattern Matching

Die meisten automatisierten E-Mail Antwort- oder Klassifizierungsprogramme arbeiten mit einem Haufen an Termen die in einem Vektormodell tf-idf gewichtet sind. Außerdem sind alle sog. *stop words* entfernt worden.
Ein *Text Pattern* welches zu einem bestimmten Teil in einen Text passt ist eine Sammlung an lexikographischen Elementen und Regeln. Es wird zwischen drei grundlegenden Elementen unterschieden. Zum einen Wörter und Wortstämme, Zahlen oder Nummern und sog. *Entity Slots*. Ein Entity Slot kann eine E-Mail-Adresse oder eine feste Nummer sein, wie ein Autokennzeichen oder eine Telefonnummer. Generell wird die Diversität nur durch die Fähigkeit , welche für die Isolation und dem Verständnis einzelner Textabschnitte zuständig ist, limitiert. Außerdem werden die Elemente in Synonym-Sets eingeteilt Bsp: Stuhl; Möbel*.

3 Ansatz und Aufbau

3.1 Feature Engineering

Ein großer Schwerpunkt der Arbeit liegt auf der Anwendung von *Feature Engineering*. Doch was ist Feature Engineering?
Grundsätzlich ist der Begriff des Feature Engineering ein informeller Ausdruck, der im Bereich des Machine Learnings verwendet wird um das Erstellen von Feature Sets zu beschreiben. Kurz gesagt ist ein Feature nur eine Information

die eventuell bei Vorhersagen nützlich sein können. Angewandt bei unserem Beispiel wäre ein Feature die Wahl des Eingangsdatums oder der Absender. Beide Informationen können bei der Vorhersage helfen. Wenn ein bestimmter Absender bei 80% der E-Mails nur Quotinganfragen stellt, wäre die Wahl des Features sehr hilfreich für das Klassifizierungstool. Der Mangel an einfachen Heuristiken um sinnvolle von nicht sinnvollen Features zu unterscheiden, macht das Feature Engineering zu einem komplexen und zeitaufwändigen Unterfangen. Dennoch wirken sich Features ungemein positiv auf die Ergebnisse von Klassifiern aus. Nur wie geht man vor und auf was muss geachtet werden?

3.2 Ansatzerläuterung

Es gibt Möglichkeiten den möglichen Featureraum einzugrenzen. Ein probates und einfachs Mittel ist die Verwendung von *Stopwordlisten*. Um einen Text semantisch zu verstehen sind viele Wörter im Text überflüssig und sagen nichts aus. Wörter wie "aber", "und" oder "ob" sind so genannte Stopwords und können entfernt werden. Das verkleinert nicht nur den Datensatz sondern entfernt auch gleich mögliche ineffiziente Features. Ein weitere wirksame Methode zur Featureelimination ist die Anwendung von *Lemmatization*. Dabei Terme (Term = Wort) gebildet welche aus Pluralen und konjugierten Verben einen einzelnen Wortstamm bildet. In unserem Fall handelt es sich bei den Datensätzen um E-Mails, welche keine langen texte beinhalten. Dadurch fällt eine *Normalization* weg. Damit ist die Wortauftrittshäufigkeit im Vergleich zur Textlänge gemeint, welche durch häufiges Auftreten eine größere Gewichtung bekommt.

4 Implementierung

5 Ausblick, Abschluss