A2, Umsetzung, Abschlussbericht

Philip Weinmann, phw8736@thi.de, Gruppe 1

25. Januar 2022

Zusammenfassung

Dieser Bericht erklärt den entwickelten Text Klassifikator. Der Klassifikator wurde durch ein Naive-Bayes-Modell umgesetzt und prognostiziert für unbekannte Texte die entsprechende Domäne. Vorweggenommen sei, dass auf dem 20 Newsgroups-Datensatz die höchste Accuracy mit dem Modell bei 81,22% liegt.

1 Auseinandersetzung mit der Problemstellung

Um einige der folgenden Probleme analysieren zu können, muss zunächst die grundlegende Funktionsweise des Naive-Bayes geklärt werden. Beim Naive Bayes werden Posterior-Wahrscheinlichkeiten durch den Satz von Bayes berechnet. Für alle in den Trainingsdaten existierenden Klassen (Outputlabels) werden anhand von Inputdaten Wahrscheinlichkeiten für die Zugehörigkeit der Daten zur Klasse berechnet. Das argmax der berechneten Wahrscheinlichkeiten ist dann die Prognose für eine Inputvariable:[Ras14]

$$P(\omega_j \mid \mathbf{x}_i) = \frac{P(\mathbf{x}_i \mid \omega_j) \cdot P(\omega_j)}{P(\mathbf{x}_i)}$$

$$\text{predicted class label} \leftarrow \underset{j=1\ldots,m}{\arg\max}\, P(\omega_j \mid \mathbf{x}_i)$$

Es ist zu erwähnen, dass die Verteilung der Texte $p(x_i)$ für die Berechnung der posterior-Wahrscheinlichkeiten $p(w_j|x_i)$ ignoriert werden kann, da die vorhergesagten posterior-Wahrscheinlichkeiten auch ohne diesen Skalierungsfaktor proportional zueinander sind, und die Vorhersage mit oder ohne dem Faktor gleich sein wird.

1.1 Textvorverarbeitung

1.1.1 Allgemeines

Um die Güte des Modells zu verbessern, sollte eine einheitliche Textvorverarbeitung über die Trainingsdaten, Testdaten und auch später die zu klassifizierenden unbekannten Daten laufen. Interessante Methoden hierzu sind *Stemming, Lemmatization, Bag-of-words, n-gramm SLM, TF-IDF,* oder auch *Stop-Word-Removal.* Ziel von *Stemming-* und Lemmatization- Methoden ist es, die Wörter des gegebenen Texts auf bestimmte Wortbestandteile zu reduzieren. Durch *n-gramm Statical Language Models* kann der Kontext von Wörtern innerhalb des Satzes oder Textes miteinbezogen werden. Ein Bag-of-words generiert ein Vokabular aller vorkommenden Wörter in einem Text und erstellt dann für jeden weiteren Text einen Vektor. Der Vektor enthält am Index des Wortes i eine 1, falls das Wort i im Text vorkommt, ansonsten eine 0. Interessant ist außerdem das *TF-IDF*-Modell, bei dem berücksichtigt wird, dass sehr frequente, dominante Wörter innerhalb eines Textes möglicherweise keinen Beitrag zur Information über die Kategorie enthalten. Stattdessen werden seltenere, aber dafür domänenspezifische Wörter mit einem höheren Score bewertet.

1.1.2 Umsetzung im eigenen Modell

Für alle Tests wurde mit dem gleichen Trainingsdatensatz auf einem *Naive Bayes*-Algorithmus trainiert und mit dem gleichen Testdatensatz getestet. Es wurde hierbei immer mit einem Bag-of-Words-Modell

gearbeitet.Aus diesem Grund wurden alle Worthäufigkeiten innerhalb einer Textkategorie, sowie alle Kategoriehäufigkeiten abgespeichert.
Wie im abgegebenen Code ersichtlich ist, wurden verschiedene Kombinationen aus folgenden Textvorverarbeitungsoptionen und Parametern getestet:

- *Stop-Word-Removal*
- *Removing of the n most frequent words in test- and trainingdata*
- *Lemmatization*
- *Stemming*
- *Additive-Smoothing* mit verschiedenen Werten für α
- *Reducing the dominance of frequent words* (durch Transformation der *word-counts*)

1.1.2.1 Transformation der *word-counts*

Insbesondere der zuletzt genannte Punkt der Transformation von word-counts ist im Vergleich zum Modell der vergangenen Woche hervorzuheben. Die *word-counts* f_i wurden folgendermaßen bearbeitet:

$$f_i = \log 2(f_i + 1)$$

Diese Veränderung sorgt dafür, dass besonders häufig auftretende Wörter innerhalb eines Textes weniger berücksichtigt werden. Das hohe Gewicht durch die Dominanz des Wortes wird reduziert und die seltener auftretenden Wörter werden stärker berücksichtigt.[MOH15] Der Fokus liegt durch diese Anpassung nun stärker auf den *"Key-Words"* der jeweiligen Textkategorie. Allein durch diese Transformation konnte eine Verbesserung der *Accuracy* von über 10% erreicht werden.

1.1.2.2 Folgen der Transformation der *word-counts*

Im Zusammenhang mit dieser Anpassung wurde auch der *word-cout* aller Wörter einer Klasse mit der gleichen Formel angepasst, so dass die Normiertheit unserer Wortwahrscheinlichkeiten verloren gegangen ist. Hierzu wurde in einem Test festgestellt, dass keine Wortwahrscheinlichkeiten mehr im Naive-Bayes-Klassifikator berechnet werden. Stattdessen kann man sich die Werte eher als Gewichte von Worten innerhalb einer Klasse vorstellen. Trotzdem liefert die Transformation der *word-count* den genannten Perfomance-Gewinn.

1.2 Laufzeit

Im ersten Zwischenbericht gab es noch einige Probleme innerhalb des Codes, so dass die Laufzeit für den Testdatensatz bei über zwei Stunden lag. Mittlerweile konnten diese Probleme extrem reduziert werden. Der Durchlauf von der Textvorverarbeitung bis hin zur Ausgabe der Konfusions-Matrix beträgt nun, bei Ausführung über *Google Collab*, maximal zwei Minuten.

2 Prognosegenauigkeit

2.1 *Accuracy* und *Matthews Correlation Coefficient*

Die Prognosegenauigkeit kann über die *Accuracy* angegeben werden, welche das Verhältnis von richtigen Prognosen zu getesteten Daten angibt. Ziel ist es die Accuracy auf den Testdaten zu maximieren. Hierzu wurden verschiedene Modelle getestet, um das Modell mit der höchsten *Accuracy* zu finden.

Getestete Modelle							
Modell k	Stopword-Removal	Stemming	Lemmatization	α in Additive Smoothing	n most frequent words removed	ACC in %	MCC in %
0	no	no	no	9	0	81.22	80.26
1	no	yes	no	1	0	72.54	71.19
2	yes	no	yes	1	0	80.73	79.74
3	yes	no	no	12	10	80.42	79.43
4	no	no	no	1	11	73.22	71.89
5	yes	yes	no	11	12	80.58	79.59
6	yes	yes	no	11	0	80.61	79.62
7	TEST	ON	TRAINING	DATA	3	95.22	94.97

Tabelle 1: Modellkonstellationen mit ACC- und MCC-Scores

2.2 Ergebnisse

Für die getesteten Modelle konnten die in der Tabelle1 dargestellten *Accuracy-Scores* und *Matthews Correlation Coefficients* erzielt werden.

3 Interpretation der Ergebnisse

Eine *Accuracy* von über 81% auf dem Testdatensatz ist im Vergleich zu 61% aus dem ersten Zwischenbericht eine deutliche Steigung und soll an dieser Stelle ausreichen. Der *MCC* kann als Zusammenfassung mehrer *Person-Koeffizienten* verstanden werden und gibt die Stärke des linearen Zusammenhangs von Variablen an. Da auch hier ein Wert von ca. 80% erzielt wird, kann dies als starker linearer Zusammenhang interpretiert werden.

4 Verbesserungsvorschläge

Wie bereits von Farah Mohamed in[MOH15] herausgefunden wurde, wird die *Accuracy* durch den Einsatz von *n-grams* noch einmal deutlich erhöht. Auch in dessen Tests wurde mit dem *20-Newsgroups*-Datensatz und einem *Naive Bayes*-Modell gearbeitet. Durch das Einfügen der *n-grams* konnte Farah Mohamed die Accuracy von 73.83% auf über 86% erhöhen.
Insbesondere sollte sich auch um das in 1.1.2.1 und 1.1.2.2 genannte Problem der verlorenen Normiertheit des Modells gekümmert werden. Da wir, wie bereits beschrieben, den *log2* sowohl unserer *word-counts* der einzelnen Wörter in einer Klasse $c(w_i|y)$, als auch der Gesamtanzahl an Wörtern innerhalb einer Klasse $c(w_{total}|y)$ nehmen und die 'Wahrscheinlichkeit' so mit der Formel
$p(x|y) = \log 2(c(w_i|y))/\log 2(c(w|y))$ berechnen, übersteigt die Summe der Wahrscheinlichkeiten den Wert 1.

5 Abschließende Anpassungen

Es wurde abschließend an den problematischen *log-wordcounts* gearbeitet. Diese wurden im finalen Code entfernt und durch log-Wahrscheinlichkeiten ersetzt. Das bedeutet, dass nicht mehr der Logarithmus einzeln auf die *word-counts* der einzelnen Wörter in einer Klasse $c(w_i|y)$ und die Gesamtanzahl an Wörtern innerhalb einer Klasse $c(w_{total}|y)$ angewendet wurde, sondern auf die gesamte Wahrscheinlichkeitsberechnung. Die Folge daraus ist eine Änderung der Formel von $p(x|y) = \log 2(c(w_i|y))/\log 2(c(w|y))$ hin zur korrekten Formel $p(x|y) = \log 2(c(w_i|y)) - \log 2(c(w|y))$.
Außerdem wurde eine *TFIDF-stop-word-list* anstelle der von *NLTK* vorgefertigten Liste erstellt. Allerdings konnten durch Filtern der Wörter aus dem Corpus mit der neuen Liste, auch unter vorheriger Anwendung des *Porter-Stemmer*, keine besseren Ergebnisse erzielt werden. (Accuracy bei ca. 79.8%)
Das beste getestete Modell mit den verbesserten *log-Likelihoods* liefert die in der folgenden Tabelle dargestellten Ergebnisse.
Die im abgegebenen Code auskommentierten Zeilen unter dem Reiter '*Testing*' dienen zur Anwendung

von *Stemming* und dem Entfernen der *stop-words*, mit einem definierbaren *treshold* für den TFIDF.

Getestete Modelle							
Modell k	Stopword-Removal	Stemming	Lemmatization	α in Additive Smoothing	TFIDF	ACC in %	MCC in %
0	no	no	no	0.01	no	81.4	80.44
1	TEST	ON	TRAINING DATA	0.01	no	98.41	98.33

Modellkonstellationen mit ACC- und MCC-Scores des besten Modells

Literatur

[MOH15] KHALIF FARAH MOHAMED. Using naive bayes and n-gram for document classification. *DEGREE PROJECT, IN COMPUTER SCIENCE , SECOND LEVEL, STOCKHOLM, SWEDEN 2015*, page 21, 2015.

[Ras14] Sebastian Raschka. Naive bayes and text classification i. *arXiv preprint arXiv:1410.5329*, page 20, 2014.