Hochschule für Technik Stuttgart

Studiengang Informatik

„Künstliche Intelligenz“,

WS2021/22

Prof. Dr. Ulrike Pado

KI-Gruppe Vino

vorgelegt von:

Mouhamad Masri

Matrikelnummer: 380038

Vilian Papazov

Matrikelnummer: 1000160

Vajos Amaranditis

Matrikelnummer: 1001217

Fahretdin Okutan

Matrikelnummer: 1002336

Abgabedatum: 18.11.2021

Inhaltsverzeichnis

.

1. **Analyse des Datensatzes**

„Sarcasm-Corpus“ besteht aus einer .arff-Datei aus Produktbewertungen der Nutzer.

Der Datensatz stammt aus Amazon.com er enthält 4 Merkmale. Eine „id“ was eine eindeutige nummerische Aufzählung für die Daten ist. Der „filename“ beschreibt den Namen der .txt-Datei in dem Format „Zahl_Zahl_String.txt“. Der eigentliche Review „token“ in ausschließlich englischer Sprache und zuletzt die Zielklasse die einem sagt ob der Review „ironisch“ oder „normal“ ist.

Die Daten wurden wahrscheinlich im Jahr 2012 erhoben zeitgleich mit der Veröffentlichung der Wissenschaftlichen Arbeit „Irony and Sarcasm: Corpus Generation and Analysis Using Crowdsourcing“ von Elena Filatova in der die „2-step procedure“ Methode für die Sammlung der Daten beschrieben wird.(Filatova 2012)

Er besteht aus insgesamt 1254 Datensätze. Wobei 817 davon als „normal“ gelten und 437 als „ironisch“.

Es soll vorhergesagt werden ob der Review „ironisch“ oder „normal“ ist.

2. **Trainingsprozess & Algorithmus**

Ursprungsdatensatz beinhaltet 4 Attribute und 1254 Instanzen.

Trainingsdatensatz - 80% - 1017 Instanzen – 4 Attribute

Entwicklungsdatensatz - 10% - 112 Instanzen – 4 Attribute

Testdatensatz - 10% - 125 Instanzen – 4 Attribute

Wir entschieden uns für den in der Vorlesung bereits gezeigten „J48“ Entscheidungsbaum-Algorithmus.

3. **Merkmalsauswahl**

Jeder von uns suchte danach ob es bereits fertige Algorithmen gibt die aus einem String Features extrahieren können und überlegten sich ebenfalls geeignete Merkmale die wir aus einem String ziehen können.

Für die Merkmale die wir manuell einfügen wollen kam dabei eine Liste mit folgenden Merkmalen heraus. Verben, Nomen, Adjektive, Adverbien, Pronomen, Länge des Strings, Anzahl der Wörter, Anzahl an Fragezeichen, Anzahl an Ausrufezeichen, Anzahl an sonstigen Symbolen(# |…).

Für das extrahieren der Verben, Nomen, Adjektive, Adverbien und Pronomen benutzten Vaio und Fahretdin die „nltk“ Python Library. (5. Categorizing and Tagging Words 2021) Das Zählen bestimmter Charaktere erfolgte mit Regular Expressions.

Für das erstellen einer neuen Arff-Datei verwendeten wir die „liacc-arff“ Python Library. (LIAC-ARFF 2021)

Vaio und Fahretdin fragten auf Reddit nach wie andere das Problem angehen würden und entdeckten das „WordEmbedding“ word2vec. (Create new attributes from string, what are good attributes? 2021) Obwohl das fertige Python-Skript in der Lage ist die Ähnlichkeit zwischen den Wörtern im Trainingsdatensatz vorherzusagen und ebenfalls in der Lage ist ein bestimmtes fehlendes Wort in einem Satz vorherzusagen (CBOW Model) entschieden wir uns es nicht in unserem Projekt zu verwenden da die Komplexität zu hoch ist.

Vilian und Laith fanden heraus das Weka eine Funktion namens „StringtoWordVector“ existiert. „stringToWordVector“ zählt dabei die Häufigkeit aller auftretenden Wörter dabei erstellt er für jede Zielklasse ein eigenes Wörterbuch und führt diese dann zusammen.

Die Parameter-Einstellungen von dem StringToWordVector-Algorithmus spielen eine wichtige Rolle für die Transfomration der Ursprungsdatensatz. Wir entschieden uns für die folgenden Einstellungen.

„TFTransform“ benutzen wir nicht, weil uns nicht die Bedeutung eines Wortes im Vergleich zu Bewertungslänge interessiert, sondern die Korrelation zwischen das Wort und unsere Zielklasse.

„attributeIndices“ bekommt den Wert “3”, so dass nur das dritte Attribut “tokens” bearbeitet wird.

„lowerCaseTokens“ Wir wollten nicht zwischen gleichen Wörter mit Groß- oder Kleinbuchstaben unterscheiden, lowerCaseTokens sollte True sein.

„wordsToKeep“ Anfangs dachten wir je mehr Wörter desto besser. Das war aber nicht immer richtig. Entscheidungsbaum-Algorithmus hat unterschiedlich gut einmal mit 100 Wörter und mehr Attribute, und mit 1000 Wörter und weniger Attribute.

„stemmer“ Keine Stemmer wurden eingesetzt, so dass wir nur mit vollständigen Wörtern arbeiten.

(Weka Tutorial 31: Document Classification 1 (Application) 2013)

3.1 Entfernen bestimmter Merkmale

Da der Algorithmus die Merkmale zur Vorhersage benutzt ist es wichtig unnötige nicht in Verbindung mit der Zielklasse bestehende Merkmale zu entfernen.

„id“ – Die Nummerische Aufzählung der Daten ist absolut irrelevant für den Algorithmus.

„filename“ – Obwohl der „filename“ eine bestimmte Syntax befolgt „Zahl_Zahl_String.txt“ konnten wir daraus keine Informationen extrahieren die für unsere Zielklasse relevant sind.

„tokens“ – Die Reviews sind fürs erstellen der Merkmale zwar notwendig, sind jedoch als eigenes Merkmal negativ.

Bestimmte Wörter die bei StringToWordVector erstellt werden wurden von uns ebenfalls gelöscht dabei haben wir uns auf unsere Intuition verlassen. Beispiele dafür sind Zahlen (0-9) oder Wörter wie „I“ oder „at“.

Jede Instanz besitzt nun 124 Attribute.

3.2 Evaluationsdurchläufe

1. StringToWordVector mit 100 WordsToKeep

   F-Wert Zielklasse ironisch: 0,415 normal: 0,662 average: 0,576

2. StringToWordVector mit 1000 WordsToKeep

   F-Wert Zielklasse ironisch: 0,582 normal: 0,767 average: 0,705

3. StringToWordVector mit 1000 WordsToKeep und Python Attribute

   F-Wert Zielklasse ironisch: 0,578 normal: 0,726 average: 0,688

4. StringToWordVector mit 100 WordsToKeep und Python Attribute

   F-Wert Zielklasse ironisch: 0,593 normal: 0,769 average: 0,708

Man erkennt das bei zuviel ausgewählten Wörter der Algorithmus ein schlechteres Ergebnis liefert. Eine Kombination aus den Python Attributen und 100 Wörter liefert die besten Ergebnisse von fast 60% ironisch und 77% normal beim Entwicklungs Datensatz.

4. **Abschlussevaluation**

Die Performanz(F-Wert) auf die ungesehenen Testdaten liegen bei:

0.530 für ironisch -> 0.766 für normal -> 0,691 für average

Der Entscheidungsbaum verwendet anfangs ausschließlich das Merkmal „Anzahl an Fragezeichen“. Danach scheint die „Anzahl an Adjektiven“ als wichtigstes Kriterium und zu guter letzt kommen die bestimmten Wörter die „StringToWordVector“ ausgewählt hat.

Der Entscheidungsbaum sieht ungefähr immer so aus: „Anzahl an Fragezeichen“ -> „Anzahl an Adjektiven“ -> „StringToWordVector“ Wörter

Für die Zielklasse „ironisch“ entspricht der F-Wert gerade einmal einem Münzwurf was für die echte Welt keine verlässliche Quelle ist. Sie bietet nicht einmal eine Hilfestellung in irgendeiner Form an.

Ein F-Wert von 0.766 für die Zielklasse „normal“ ist relativ aussagekräftig.

5. **Literaturverzeichnis**

5. Categorizing and Tagging Words (o. D.): Nltk, [online] https://www.nltk.org/book/ch05.html [abgerufen am 18.11.2021]

Create new attributes from string, what are good attributes? (2021): reddit, [online] https://www.reddit.com/r/learnmachinelearning/comments/qesikw/create_new_attributes_from_string_what_are_good/ [abgerufen am 18.11.2021].

Filatova, Elena (2012): Irony and Sarcasm: Corpus Generation and Analysis Using Crowdsourcing, lrec-conf, [online] http://www.lrec-conf.org/proceedings/lrec2012/pdf/661_Paper.pdf [abgerufen am 18.11.2021].

LIAC-ARFF v2.1 — liac-arff 2.0 documentation (o. D.): Pythonhosted, [online] https://pythonhosted.org/liac-arff/ [abgerufen am 18.11.2021].

Weka Tutorial 31: Document Classification 1 (Application) (2013): YouTube, [online] https://www.youtube.com/watch?v=jSZ9jQy1sfE&t=202s [abgerufen am 18.11.2021].