1 Die grosse Idee

Es gibt 2 Parteien (z.B. Admins von Mailservern in Firmen), die annehmen, dass mit einer grösseren Anzahl an Emails ein Klassifikationssystem für Spam besser trainieren kann. Deswegen wollen sie ihre bereits klassifizierten Emails gemeinsam verwenden, um so aus einer grösseren Wissensbasis einen bessere Klassifikator zu erzeugen. Das Problem dabei ist, dass keine der beiden Parteien etwas über die Emails der anderen Partei lernen darf. Deswegen ergebn sich zwei weitere Fragen:

- Wie kann der ID3-Algorithmus privat und verteilt berechnet werden?
- Wie kann der ID3-Algorithmus verwendet werden, um Emails zu klassifizieren?

2 Private Verteilung des ID3-Algorithmus

Der Private $ID3_\delta$-Algorithmus kann im ausgeteilten Paper auf Seite 192 gefunden werden. Wir sehen es als nicht sinnvoll an, diesen Algorithmus nun hier erneut aufzuführen. Daraus folgt, dass folgende Prinzipien und Protokolle weiter untersucht werden müssen:

- Yaos Protokoll, angewendet für die maximale Summe, Gleichheit und die Maximum-Gain-Bestimmung
- das '$x \cdot \ln x$'-Protokoll

3 Spam-Klassifikation des ID3-Algorithmus

Um mit dem ID3-Algorithmus Spam zu klassifizieren, ist es notwendig, Attribute zu finden und mit diesen Attributen dann einen Entscheidungsbaum mit dem Ergebnissen,,Spam" und,,kein Spam" zu konstruieren.

Ausgehend von,,Learning Spam" Ergeben sich unter anderem folgende Features:

1. Anzahl Vorkommnisse einzelner Wörter hardkodiert: Diese Variante ist auf jeden Fall einfach zu implementieren, hat jedoch das Problem, dass sie einfach zu umgehen ist (es werden entweder Synonyme verwendet oder beispielsweise Vokale vervielfacht).

2. Verwendung aller Wörter, die im Mail-Text vorkommen (nach einer Reduktion des Nachrichteninhaltes um Füllwörter): Dies sollte nicht signifikant komplizierter zu implementieren sein wie Variante 1 und liefert laut ,,Learning Spam" relativ gute Ergebnisse.

3. Zusammenfassung ähnlicher Wörter: Variante 2 hatte das Problem, dass in einer Email nur wenige dieser,,high-gain"-Wörter vorkommen, sodass die resultierenden Trainingsvektoren sehr leer waren. Um das zu umgehen, wurden Wörter anhand von Bedeutungen zusammengefasst.

3.1 Bewertung

Da das Ziel der Arbeit nicht ist, einen guten Spam-Klassifikator zu schreiben, sondern die Aspekte der verteilten, privaten Berechnugen im Fokus stehen, denkn wir, dass die erste oder die zweite Variante vorzuziehen sind, da sie beide zumindest naiv einfach zu implementieren sind und zumindest Variante 2 dann relativ brauchbare Ergebnisse liefert. In einer zweiten Ueberlegung ergibt sich jedoch, dass Variante 2 unter den Apekten der privaten Berechnung problematisch ist: Man nehme einfach mal an, dass einer der beiden Admins in einer Forschungsabteilung arbeitet, und es werden mehrere wichtige Stichpunkte geheimer Forschungsergebnisse als high-gain-Worte erkannt, die Mails sofort als 'kein Spam' klassifizieren. Daher denken wir, es sollte für eine erste Implementierung der Arbeit ein einfaches Zählen der Vorkomnisse von Worten ausreichen, wobei die Menge der interresanten Worte beispielsweise durch eine Konfigurationsdatei eingegeben werden kann. Damit muss jedoch zuvorderst noch ein sicherer Algorithmus gefunden werden, der nachprüft, dass beide Benutzer die gleiche Wortliste verwenden. Es verbleibt zu entscheiden, was passiert, wenn diese Wortliste nicht gleich sind. Wir denken jedoch, dass in diesem Fall der Vorgang abgebrochen werden sollte, da zwei ehrliche Benutzer sich vorher auf die gleiche Wortliste einigen und dann die gleiche Wortliste verwenden und deswegen Unterschiede in der Wortliste auf einen unehrlichen Anwender hindeuten.

3.2 Behandlung des kontinuierlichen Attributes,,Wortanzahl"

Ein relativ einfacher Weg, kontinuierliche Attribute zu diskretisieren ist eine einfache Fuzzifizierung. Wir schlage deswegen vor, dass die Anzahl Vorkommnisse eines Wortes durch die Gesamtanzahl Worte dividiert wird, und dann

folgende Abbildung verwendet wird:

- Falls dieser Wert $<$ *s ist, ist der Attributwert für das Wort W,,selten"*
- *Falls dieser Wert* $>$ h ist, ist der Attributwert für das Wort W,,häufig"
- sonst ist der Attributwert für das Wort W,,normal"

Die Schwellwerte *s und* h müssen dann noch festgelegt werden und sollten ebenfalls über eine Konfigurationsdatei festlegbar sein.

4 Weiteres Vorgehen

Damit ergibt sich als weiteres Vorgehen, folgende Algorithmen zu verstehen und durch kryptographische Primitive zu implementieren und dann als Vorbereitung der Implementierung aufbereitet aufzuschreiben:

- ein sicheres Protokoll, welches feststellt, ob die gleiche Wortliste Verwendet wird (vmtl ebenfalls ueber Yaos Protokoll)
- Yaos Protokoll, angewendet für die maximale Summe, Gleichheit und die Maximum-Gain-Bestimmung
- das '$x \cdot \ln(x)$'-Protokoll

5 Referenzen

- Ausgeteiltes Paper
- Learning Spam http://www.usenix.org/events/usenix03/tech/freenix03/full_papers/massey/massey_html/spam.html