1 Lerner

Requirement 1: Das Programm liest die E-Mails aus dem Verzeichnis, das der Benutzer angegeben hat, ein und behandelt sie entsprechend der Unterverzeichnisse als Spam oder Nicht Spam
Requirement 2: Wenn der Anwender keine Aussage über die Ausgabeform des Programmes trifft, wird der Klassifikator auf die Standardausgabe geschrieben

Requirement 3: Wenn der Anwender mit einem Kommandozeilenparameter einen Dateinamen für die Ausgabe definiert, wird der Klassifikator in diese Ausgabedatei geschrieben.

Requirement 4: Wenn der Anwender mit einem Kommandozeilenparameter einen Dateinamen für die Ausgabe definiert und dieser Dateiname ist '-', dann wird der Klassifikator auf die Standardausgabe geschrieben

Requirement 5: Wenn der Anwender den Ausgabeparameter mehrfach angibt, gibt die Anwendung einen Fehler aus

Requirement 6: Die Anwendung kann mit einem Parameter `--server` gestartet werden. Zusätzlich muss ein Parameter `--port=INTEGER` angegeben werden. Dies veranlasst die Anwendung, einen Server zu starten, welcher auf dem Rechner auf dem angegebenen Port auf den Client wartet. Falls hier ein Fehler auftritt (beispielsweise bei Berechtigungsproblemen oder bei bereits belegten Ports) bricht der Server mit einer entsprechenden Fehlermeldung ab
Requirement 7: Die Anwendung kann mit einem Parameter `--client` gestartet werden. Zusätzlich müssen dann die Parameter `--port=INTEGER` und `--server-ip=IP-Addresse` angegeben sein. Die Anwendung versucht dann, eine Verbindung zum Server unter der gegebenen IP und dem gegebenen Port aufzubauen und den Lernvorgang durchzuführen. Sowohl Client als auch Server beenden sich nach einem Lernvorgang. Sollte beim Verbinden ein Problem auftreten (Unerreichbarkeit des Servers, Port geschlossen), so beendet sich der Client mit einer entsprechenden Fehlermeldung

Requirement 8: Wenn die Spam-Mails die Mails "Foo Bar", "Foo Bar", "Foo Foo" und "Foo" sind und die Nicht-Spam-Mails "Bar Bar", "Foo" und "Bar", dann muss diese Phase die folgende Tabelle berechnen:

Wort	Spam-Anteil	Nicht-Spam-Anteil
Foo	$\frac{5}{7}$	$\frac{1}{4}$
Bar	$\frac{2}{7}$	$\frac{3}{4}$

Eine Approximation der Werte durch Fliesskommazahlen ist ebenfalls akzeptabel.
Requirement 9: Wenn N = 2 ist, und als Worte mit Vorkomnissen gegeben sind:

Wort	Vorkomnisse in Spam-E-Mails	Vorkommnisse in Nicht-Spam-Emails
A	0.5	0.5
B	0.2	0.2
C	0.3	0.5
D	0.2	0.8
E	0.9	0.2

dann werden als Wortliste D und E gewählt.
Requirement 10: Wenn Alice die Wortliste A, B, C berechnet hat, und Bob die Wortliste C, D, E berechnet hat, dann muss die zusammengefasste Wortliste A, B, C, D, E sein.

Requirement 11: Das Protokoll, in dem beide Anwender ihre Wortlisten einander zusenden und lokal die spezifizierte Syncronisierung durchführen, ist implementiert
Requirement 12: Wenn die E-Mails "A A A", "A B B", "A C C" und "A A C" sind, dann muss die Software als eigenen Mittelwert für das Wort A $\frac{1}{4} \cdot (1 + \frac{1}{3} + \frac{1}{3} + \frac{2}{3}) = \frac{7}{12}$ bestimmen.
Requirement 13: Wenn a = 0.2 ist und b = 0.4, dann ist das Schwellwertpaar (0.2, 0.4)

Requirement 14: Wenn a = 0.5 ist und b = 0.4, dann ist das Schwellwertpaar (0,4, 0.5)

Requirement 15: Wenn a = b = 0.5 ist, dann ist das resultierende Schwellwertpaar (0.5, 0.5)

Requirement 16: Das Protokoll, indem beide Anwender ihre eigenen Schwellwerte an den jeweils anderen Anwender senden und dann die bereits spezifizierte Merge-Operation ausführen ist implementiert.
Requirement 17: Wenn der E-Mail-Inhalt "A A A A B B C D" ist und die Attribute sind (A, 0.2, 0.3), (B, 0.1, 0.9), (C, 0.5, 0.8), dann wird diese E-Mail diskretisiert in den Vektor oft, mittel, selten
Requirement 18: Dieser Schaltkeris für die dominierende Ausgabe kann generiert werden
Requirement 19: Dieser Schaltkreis zur Bestimmung der dominierenden Ausgabe kann generiert werden.
Requirement 20: Es ist möglich, einen Schaltkreis so zu erweitern (separate Ausgaben)

Requirement 21: Es ist möglich, einen Schaltkreis so zu erweitern (Shares)
Requirement 22: Dieser Schaltkreis zur Berechnung der Shares der ersten Approximation kann generiert werden
Requirement 23: Dieses Protokoll zur Polynomauswertung kann ausgeführt werden.

Requirement 24: Dieses Protokoll zum 1-out-of-N-Oblivious Transfer kann ausgeführt werden.

Requirement 25: Es ist so eine einweg Funktionsfamilie implementiert.
Requirement 26: Dieses Protokoll zur Multiplikation kann ausgeführt werden

Requirement 27: Die Entropie-Shares können anhand diesse Protokolles berechnet werden
Requirement 28: Dieser Vergleichsschaltkreis kann generiert werden.
Requirement 29: Dieses Protokoll kann implementiert werden
Nouns:

- Programm, Anwendung, Software
- E-Mails
- Verzeichnis
- Benutzer, Anwender
- Unterverzeichnis
- Spam, Spam-Mails
- Nicht Spam, Nicht-Spam-Mails
- Ausgabeform
- Klassifikator
- Standardausgabe
- Kommandozeilenparameter, Parameter
- Dateiname
- Ausgabedatei
- Fehler, Problem
- Server
- Client
- Berechtigungsproblem
- Port
- Fehlermeldung
- IP Adresse, IP
- Verbindung
- Lernvorgang
- Unerreichbarkeit
- Phase
- Tabelle
- Spam-Anteil
- Nicht-Spam-Anteil
- Approximation
- Wert
- Fliesskommazahl
- Wort
- Vorkomniss
- Wortliste
- Protokoll
- Syncronisierung
- Mittelwert
- Schwellwertpaar
- Schwellwert
- Merge-Operation
- E-Mail Inhalt
- Attribut
- Vektor
- Schaltkreis
- Ausgabe
- Share, Entropie-Share
- Polynomauswertung
- Polynom
- 1-out-of-N-Oblivious-Transfer
- FunktionsFamilie
- Multiplikation
- Vergleichsschaltkreis

2 Klassifikator

Requirement 30: Der Klassifizierer weist die Eingabe Decide((Foo, 0.5,2), Output(Spam), Output(Spam)) wegen eines zu grossen Schwellwertes zurück
Requirement 31: Der Klassifizierer weist die Eingabe Decide((Foo, 2, 3), Output(Spam), Output(Spam)) wegen eines zu grossen Schwellwertes zurück
Requirement 32: Der Klassifizierer weist die Eingabe Decide((Foo, 1, 0), Output(Spam), Output(Spam)) wegen falsch sortierter Schwellwerte zurück
Requirement 33: Der Klassifizierer weist die Eingabe Decide((Foo, 0.2, 0.3), Output(Spam), Output(Spam)) wegen zuweniger Teilbäume zurück
Requirement 34: Der Klassifizierer weist die Eingabe Decide((Foo, 0.2, 0.3), Output(Spam), Output(Spam), Output(Spam), Output(Spam)) wegen zuvieler Teilbäume zurück
Requirement 35: Der Klassifizierer akzeptiert die Eingabe Decide((Foo, 0.2, 0.3), Output(Spam), Decide((Bar, 0.3, 0.4) Output(Spam), Output(Not Spam), Output(Spam)), Output(Spam))
Requirement 36: Der Klassifizierer akzeptiert die Eingabe Decide((Foo, 0, 0.5), Output(Spam), Output(Spam))
Requirement 37: Der Klassifizierer akzeptiert die Eingabe Decide((Foo, 0.5, 1), Output(Spam), Output(Spam))
Requirement 38: Der Klassifizierer akzeptiert die Eingabe Decide((Foo, 0.5, 0.5)), Output(Spam), Output(Spam))
Requirement 39: Der Klassifizierer akzeptiert die Eingabe Decide((Foo, 0, 0), Output(Spam))
Requirement 40: Der Klassifizierer akzeptiert die Eingabe Decide((Foo, 1, 1), Output(Spam))
Requirement 41: Gegeben der Klassifikator Decision((Bar, 0.3, 0.6), Output(Spam), Output(Not Spam), Output(Spam)), dann wird die E-Mail "Bar Foo Foo Foo" als Spam klassifiziert
Requirement 42: Gegeben der Klassifikator Decision((Bar, 0.3, 0.6), Output(Spam), Output(Not Spam), Output(Spam)), dann wird die E-Mail "Bar, Bar, Foo, Foo" als Not Spam klassifiziert
Requirement 43: Gegeben der Klassifikator Decision((Bar, 0.3, 0.6), Output(Spam), Output(Not Spam), Output(Spam)), dann wird die E-Mail "Bar, Bar, Bar, Foo" als Spam klassifiziert
Requirement 44: Gegen der Klassifikator Decision((Bar, 0.3, 0.6), Output(Spam), Output(Not Spam), Output(Spam)) und eine Verzeichnisstruktur wie oben skizziert, wobei hank/mail1 "Bar Foo Foo Foo", hank/mail2 "Bar Bar Foo Foo" und bob/mail1 "Bar Bar Bar Foo" enthält, dann wird die oben als Beispiel genannte Ausgabe produziert (oder in einer anderen Reihenfolge

- Klassifizierer
- Eingabe
- Schwellwert
- Teilbaum
- E-Mail
- Spam
- Nicht Spam
- Verzeichnisstruktur
- Beispiel
- Ausgabe