Todo list

Contents

1 Einleitung

Struktur vom Dokument erläutern

1.1 Begriffe

Definition: Eigenes,Gesamtes
M sei eine Menge von Elementen, die in zwei Teilmengen M_A und M_B zerfällt, sodass $M = M_A \cup M_B$ ist. Wir nehmen desweiteren an, dass Alice M_A kennt, aber weder M noch M_B und dass foo Bob M_B kennt, aber weder M noch M_A. Dann bezeichnen wir:

- M als **gesamtes** Wissen
- M_A als das **eigene** Wissen von Alice
- M_B als das **eigene** Wissen von Bob
- M_B als das **andere** Wissen von Alice
- M_A als das **andere** Wissen von Bob

Definition: Gemeinsam
Wenn beide Anwender das gleiche Wissen w haben, dann bezeichnen wir w als **gemeinsames Wissen** .
Definition: Vorwissen
Wenn Anwender verschiedene Phasen hintereinander ausführen, dann bezeichnen wir das Wissen aus den bereits ausgeführten Phasen als **Vorwissen** .

Definition: Entscheidungsbaum

Definition: Attribut
Wir definieren eine Menge von Wahrscheinlichkeiten $P = [0, 1] \subset \mathbb{R}$ und eine Menge von Buchstaben $\Sigma = \{a, b, \ldots, z, A, B, \ldots, Z\}$. Damit definieren wir ein **Attribut** als $\Sigma^+ \times P \times P$. Wenn ein Attribut $A = (w, l, h)$ gegeben ist, bezeichnen wir w als **Wort** , l als **unterer Schwellwert** und h als **oberer Schwellwert** . Es wird desweiteren von allen Attributen gefordert, dass $l \leq h$ ist.

Definition: Schaltkreis

Definition: Entstellter Schaltkreis

1.2 Annahmen

Annahme: ehrliche anwender := "handeln nach protokoll"

2 Grundlagen der Anwendung

Vision der Anwendung

2.1 Form der Benutzereingabe

Festlegen, wie die E-Mails ins Programm kommen

2.2 Interaktion der verteilten Programme

Festlegen, wie das Programm verteilt wird und die Teile kommunizieren

2.3 Phasen der Anwendung

Kurz die Einzelnen phasen der Anwendung beschreiben

3 Finden der gemeinsamen Wortliste

Einleitung, Verweisen auf Figure für gemeinsame Wortliste

Für section-Titel besseren Begriff für "Vorkomnisse der Worte in eigenen Spam/Nicht Spam E-Mails" finden

3.1 Berechnung der Vorkomnisse

Content

3.2 Sortierung der Worte nach Informationsheuristik

Content

3.3 Syncronisierung der Wortlisten

Content

4 Finden der gemeinsamen Schwellwerte

Einleitung, auf Figure für Schwellwerte verweisen

Für section-Titel besseren Begriff für "Vorkomnisse der Worte in eigenen Spam/Nicht Spam E-Mails" finden

4.1 Berechnung der Vorkomnisse

Content

4.2 Bestimmung der eigenen Schwellwerte

Content

4.3 Syncronisierung der Schwellwerte

Content

5 Diskretisieren der eigenen E-Mails

Einleitung, Figure referenzieren

Content

6 Lernen der gesamten E-Mails

verteiltes ID3 beschreiben

6.1 Yaos Protokoll

Yaos algorithmus grundlegend zusammenfassen (garbled decision table, garbled gate, garbled circuit)

Verschlüsselung für die garbled Circuits festlegen (RSA?)

1-2 Oblivious Transfer festlegen (mit RSA?)

Feststellung der benoetigten Bytezahl beschreiben

6.2 Feststellen der dominierenden Ausgabe

Schaltkreis designen: Maximum von Summen von positiven Zahlensequenzen

6.3 Feststellen ob Ausgabe eindeutig

Schaltkreis designen: Gleichheit.

6.4 Das Entropien-Protokoll

Schaltkreis für x * log x -Protokoll aus dem Paper zusammenfassen

6.5 Attribut mit maximalem Informationsgewinn finden

Vorgheen zusammenfassen, Schaltkreis aus dominierender Ausgabe wiederverwenden

7 Verwenden des Klassifikators

Es muss zusätzlich zum Lern-Programm ein Programm geschrieben werden, welches den Klassifikator auf eine Menge von E-Mails anwendet. Wir bieten hierzu ein Programm an, welches den Klassifikator in einem einfachen Textformat einliest und diesen auf eine Menge von E-Mails anwendet. Diese Menge von E-Mails ist als Dateien in einem Verzeichnis gegeben und für jeden Dateinamen wird eine Klassifikation als Spam oder Not Spam ausgegeben.

7.1 Eingabe des Klassifikators

Ausgehend von den Definitionen eines Attributes und eines Entscheidungsbaumes kann leicht eine Grammatik erzeugt werden, welche die Form des eingegebenen Klassifikators eindeutig bestimmt. Wir notieren diese Grammatik in BNF. Diese ist so gewählt, dass sie eine LL(1)-Grammatik ist, d.h., sie ist besonders einfach zu parsen.

```
tree -> ‘Decide’ ‘(’ attribute (‘,’ tree)+ ‘)’
      | ‘Output’ ‘(’ class ‘)’.
class -> ‘Spam’ | ‘Not Spam’.
attribute -> ‘(’ word ‘,’ probability ‘,’ probabilty ‘)’.
word ->   (‘a’ | ‘b’ | ... | ‘z’ | ‘A’ | ... | ‘Z’)+
probability -> number ‘.’ number .
number -> (‘0’ | ‘1’ | ... | ‘9’)+.
```

Somit wäre ein kodierter Klassifikationsbaum beispielsweise:

```
Decide((Foo, 0.3, 0.6),
        Output(Spam),
        Decide((Bar, 0.5, 0.6),
                Output(Spam),
                Output(Not Spam)
              )
      )
```

Die Produktion ,,probability" beschreibt eine Zahl, die sich als eine Folge von Ziffern vor einem Dezimalpunkt und einer Folge von Ziffern nach dem Dezimalpunkt beschreiben lassen. Dies sind alle reellen Zahlen, und da wir keine komplexen Zahlen betrachten, sondern nur Verhältnisse von natürlichen Zahlen zueinander ist diese Darstellung ausreichend, um alle möglichen Zahlenwerte darzustellen. Da die Buchstabenmenge als die lateinischen Klein- und Grossbuchstaben definiert ist und ein Wort definiert ist als Sequenz dieser Zeichen, ist die Produktion ,,word" ausreichend, um alle möglichen Worte darzustellen. Damit ergibt sich, dass die Produktion ,,attribute" in der Lage ist, alle Attribute, die in dieser Anwendung auftreten können, darzustellen.
Es gibt desweiteren bei uns nur die Ausgaben ,,Spam" und ,,Nicht Spam". Damit ist die Produktion ,,class" ausreichend, um alle möglichen Ausgaben

des Baumes darzustellen. Daraus folgt, dass die zweite Produktion von tree in der Lage ist, alle möglichen Blätter darzustellen. Desweiteren folgt induktiv aus der Vollständigkeit der Produktion „attribute" und der Darstellbarkeit der Blätter als Induktionsanfang, dass die erste Produktion von tree in der Lage ist, alle möglichen auszugebenden Entscheidungsbäume darzustellen. Damit ist die Grammatik mächtig genug für unsere Zwecke.
Es ist weiterhin zu bemerken, dass eine semantische Validierung notwendig ist, da in der Grammatik weder gefordert ist, dass der untere Schwellwert eines Attributes wirklich kleiner ist als der obere Schwellwert eines Attributes noch dass beide Schwellwerte zwischen 0 und 1 liegen, noch dass die Anzahl der Teilbäume richtig ist. Damit ergeben sich die folgenden Akzeptanzkriterien:
Requirement 1: Der Klassifizierer weist die Eingabe Decide((Foo, 0.5,2), Output(Spam), Output(Spam)) wegen eines zu grossen Schwellwertes zurück
Requirement 2: Der Klassifizierer weist die Eingabe Decide((Foo, 2, 3), Output(Spam), Output(Spam)) wegen eines zu grossen Schwellwertes zurück
Requirement 3: Der Klassifizierer weist die Eingabe Decide((Foo, 1, 0), Output(Spam), Output(Spam)) wegen falsch sortierter Schwellwerte zurück
Requirement 4: Der Klassifizierer weist die Eingabe Decide((Foo, 0.2, 0.3), Output(Spam), Output(Spam)) wegen zuweniger Teilbäume zurück
Requirement 5: Der Klassifizierer weist die Eingabe Decide((Foo, 0.2, 0.3), Output(Spam), Output(Spam), Output(Spam), Output(Spam)) wegen zuvieler Teilbäume zurück
Requirement 6: 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 7: Der Klassifizierer akzeptiert die Eingabe Decide((Foo, 0, 0.5), Output(Spam), Output(Spam))
Requirement 8: Der Klassifizierer akzeptiert die Eingabe Decide((Foo, 0.5, 1), Output(Spam), Output(Spam))
Requirement 9: Der Klassifizierer akzeptiert die Eingabe Decide((Foo, 0.5, 0.5)), Output(Spam), Output(Spam))
Requirement 10: Der Klassifizierer akzeptiert die Eingabe Decide((Foo, 0, 0), Output(Spam))
Requirement 11: Der Klassifizierer akzeptiert die Eingabe Decide((Foo, 1, 1), Output(Spam))

7.2 Arbeitsweise des Klassifikators

Der Klassifikator bekommt als Eingabe ein Verzeichnis mit E-Mails und eine Datei meinem einem Klassifikator. Den Klassifikator liest er ein und speichert ihn intern. Danach traversiert der Klassifikator das gegebene Verzeichnis rekursiv und behandelt jede Datei, die er in diesem Verzeichnis findet als E-Mail-Inhalt. (Dadurch ist es möglich, die Trainingsdaten auch als Versuchsdaten zu benutzen, ohne sie zu bewegen).
Für jede E-Mail wird dann der Inhalt der E-Mail eingelesen und die Klassi-

fikation durch den eingegebenen Klassifikator durchgeführt, d.h., für jeden Ast werden die Vorkommnisse des Wortes im Attribut festgestellt und rekursiv im entsprechenden Teilbaum weiterklassifiziert und in einem Blatt wird die Klasse festgestellt. Diese festgestellte Klasse wird dann zusammen mit dem relativen Pfad vom eingegebenen Verzeichnis ausgegeben.
Wenn also beispielsweise folgende Verzeichnisstruktur gegeben ist:

```
mails/hank/mail1
mails/hank/mail2
mails/bob/mail1
```

und wir annehmen, dass der eingegebene Klassifikator E-Mails mit dem Index 1 als Spam erkennt und E-Mails mit dem Index 2 als Nicht Spam, dann wäre die Ausgabe:

```
hank/mail1 Spam
hank/mail2 Not Spam
bob/mail1 Spam
```

Damit ergeben sich folgende Akzeptanzkriterien:
Requirement 12: 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 13: 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 14: 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 15: 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

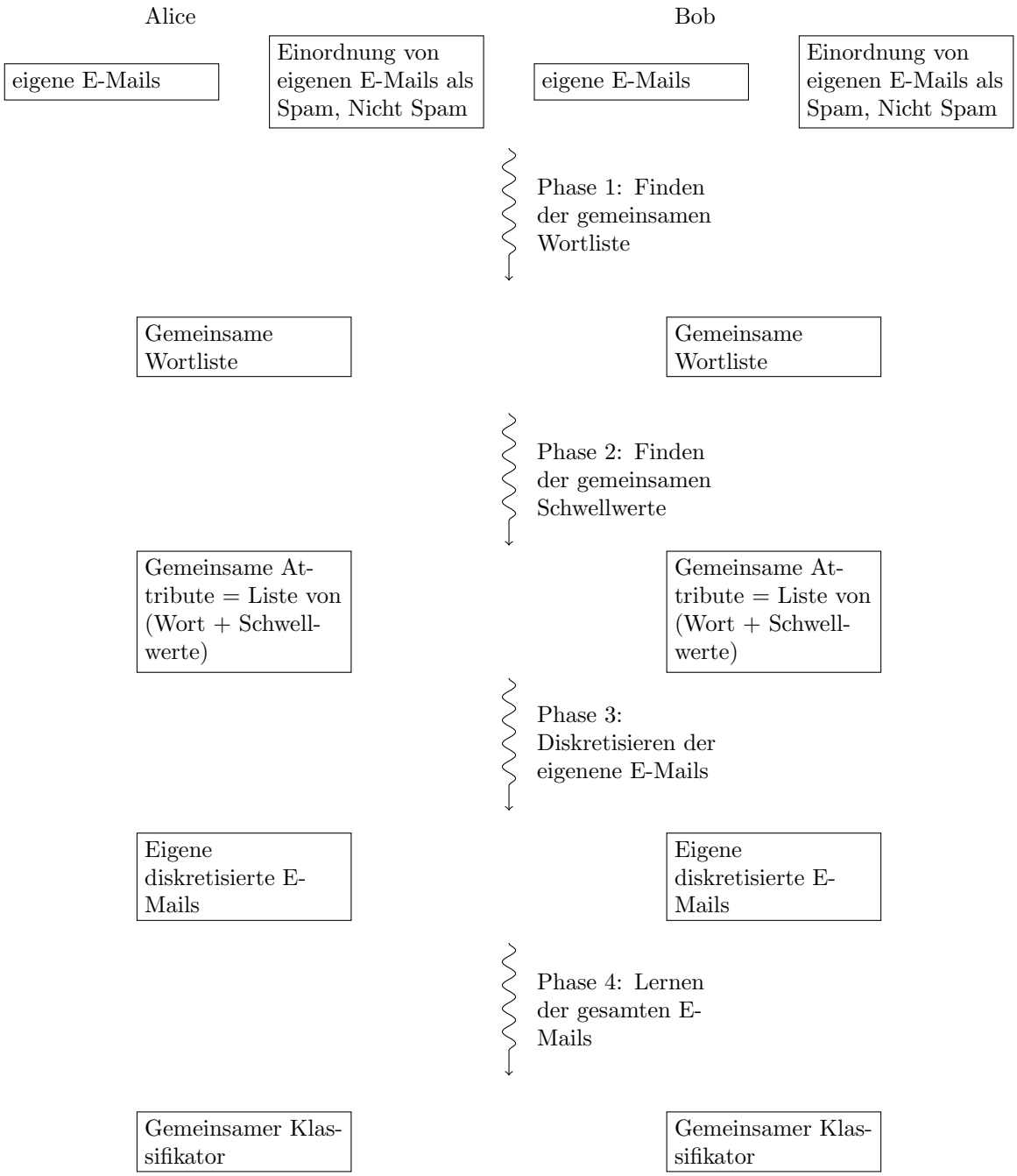


Figure 1: Phasen der Anwendung

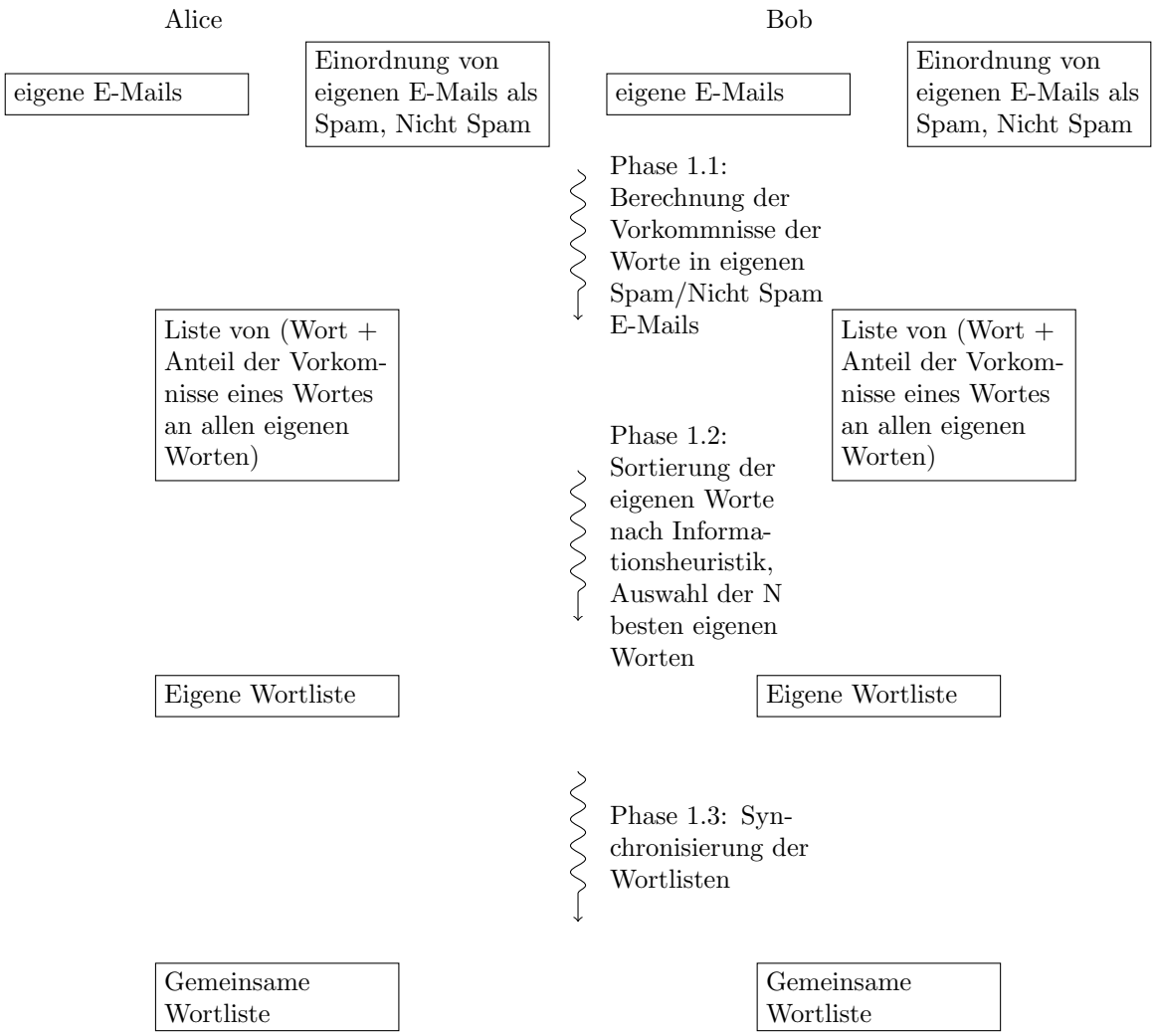
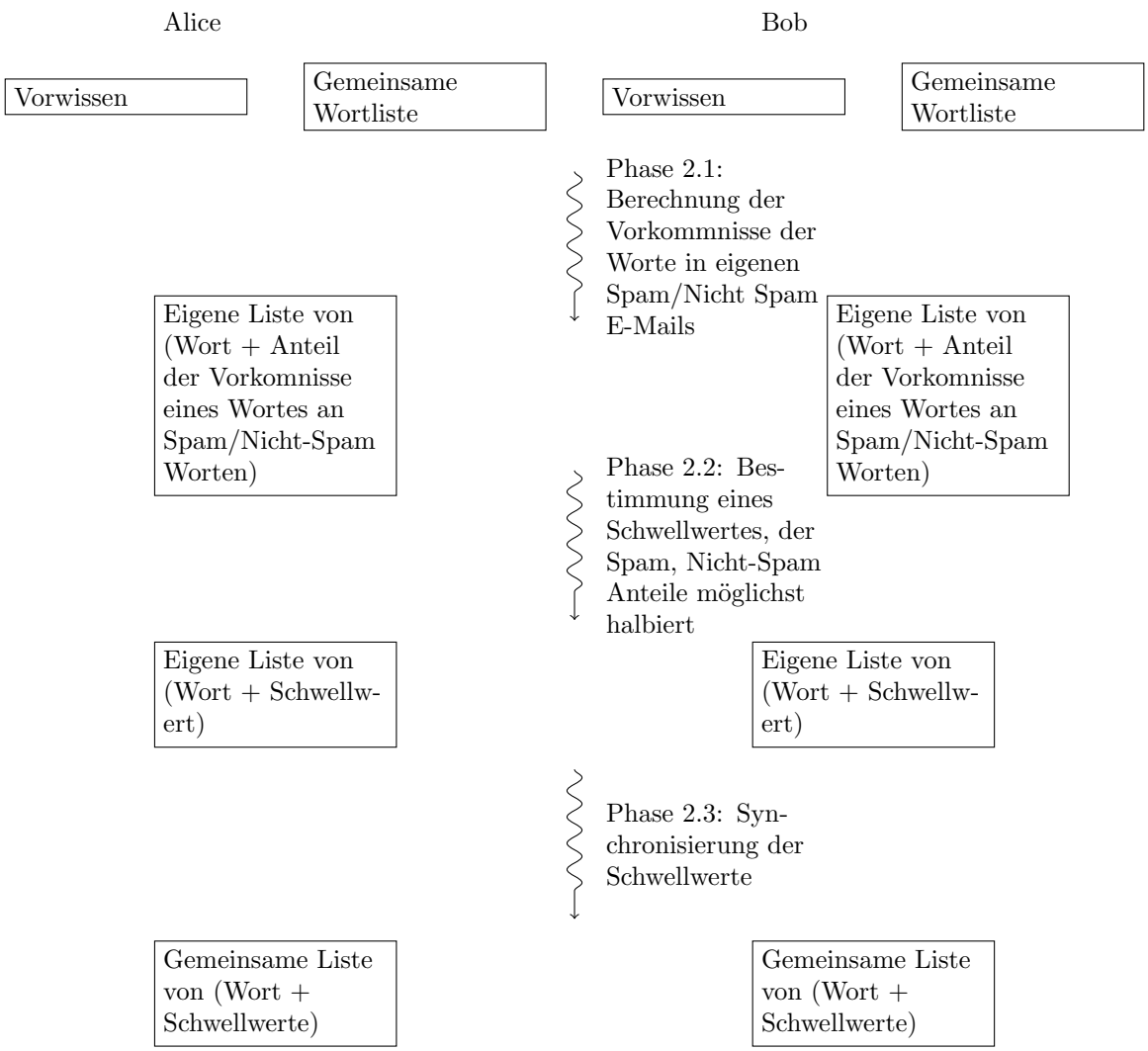
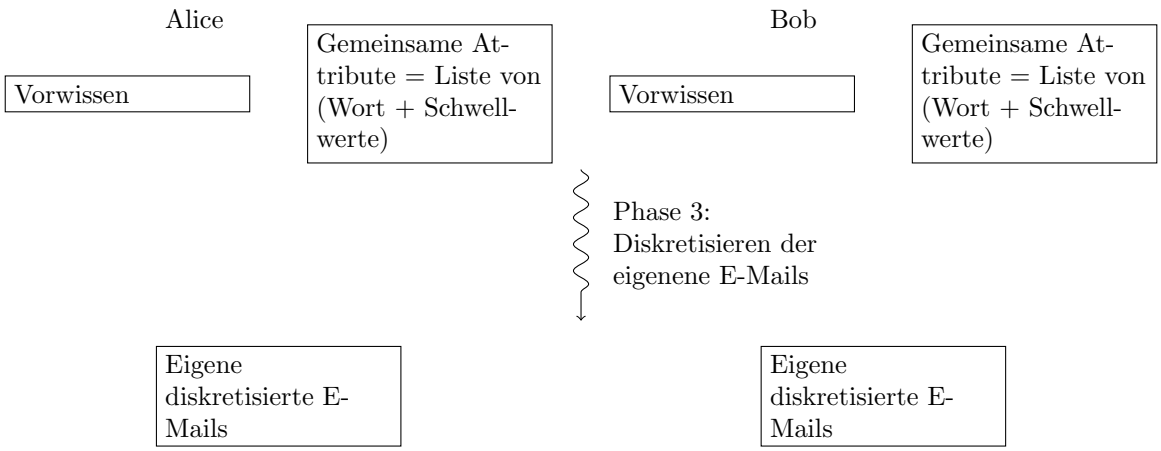


Figure 2: Schritte zum Berechnen der gemeinsamen Wortliste


Alice
Bob
Vorwissen
Gemeinsame Wortliste
Vorwissen
Gemeinsame Wortliste
Phase 2.1: Berechnung der Vorkommnisse der Worte in eigenen Spam/Nicht Spam E-Mails
Eigene Liste von (Wort + Anteil der Vorkomnisse eines Wortes an Spam/Nicht-Spam Worten)
Eigene Liste von (Wort + Anteil der Vorkomnisse eines Wortes an Spam/Nicht-Spam Worten)
Phase 2.2: Bestimmung eines Schwellwertes, der Spam, Nicht-Spam Anteile möglichst halbiert
Eigene Liste von (Wort + Schwellwert)
Eigene Liste von (Wort + Schwellwert)
Phase 2.3: Synchronisierung der Schwellwerte
Gemeinsame Liste von (Wort + Schwellwerte)
Gemeinsame Liste von (Wort + Schwellwerte)


Alice
Vorwissen
Gemeinsame Attribute = Liste von (Wort + Schwellwerte)
Bob
Vorwissen
Gemeinsame Attribute = Liste von (Wort + Schwellwerte)
Phase 3: Diskretisieren der eigenene E-Mails
Eigene diskretisierte E-Mails
Eigene diskretisierte E-Mails

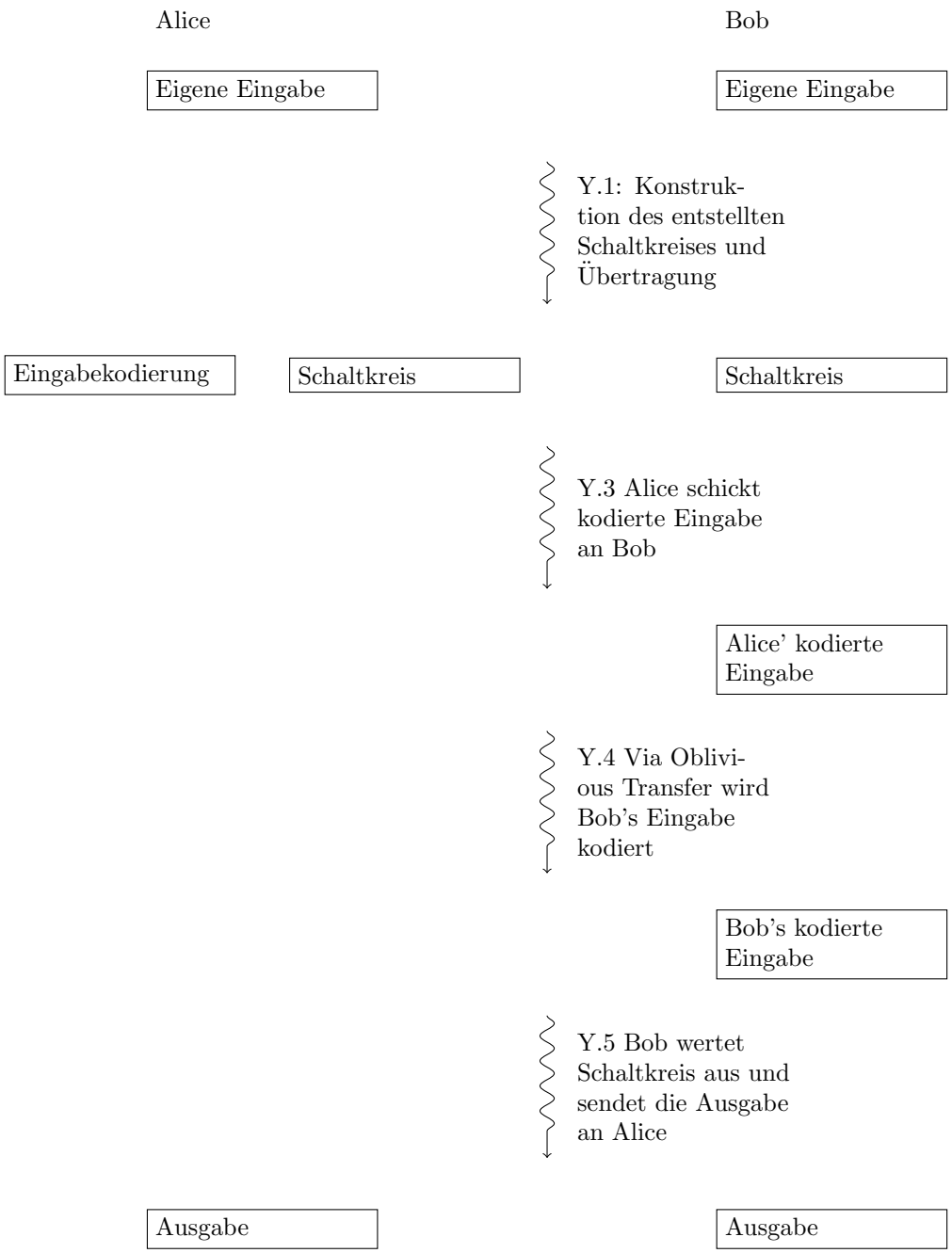


Figure 3: Yaos Algorithmus

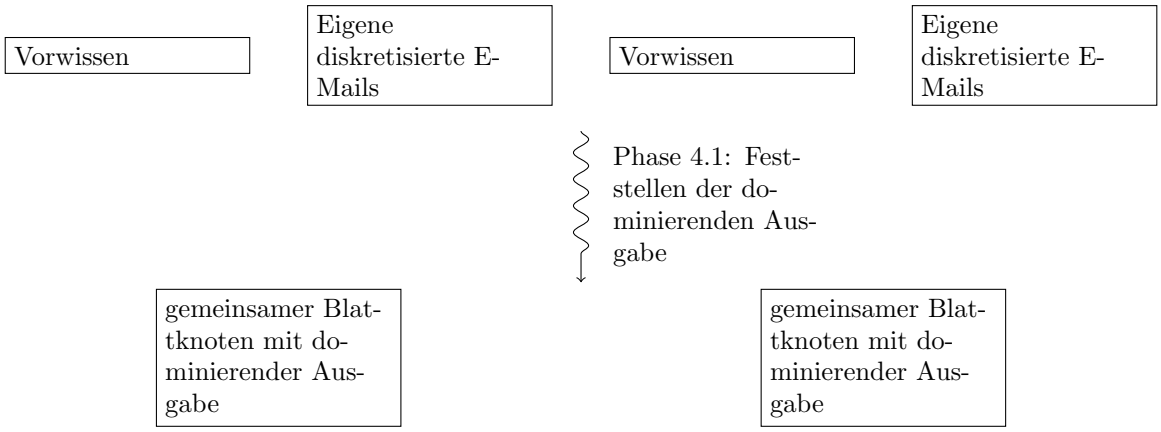


Figure 4: ID3-Algorithmus, Fall 1: Keine Attribute mehr vorhanden

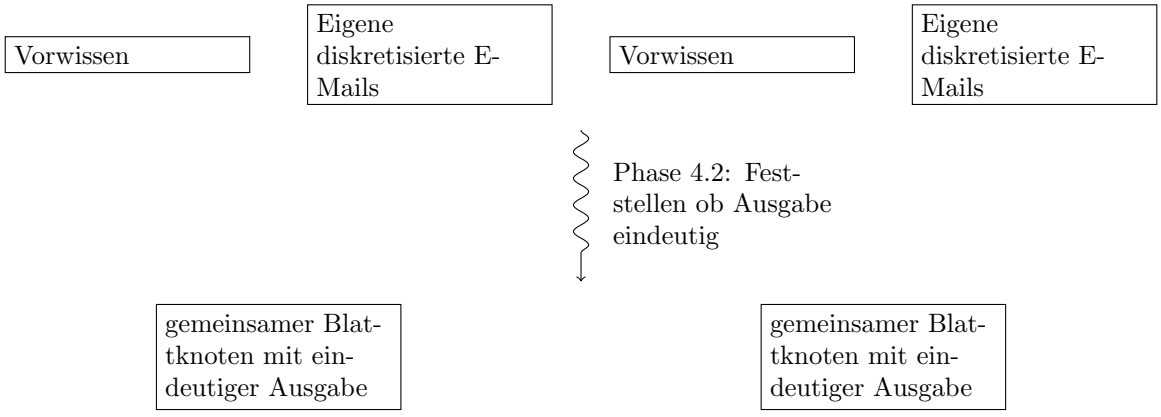


Figure 5: ID3-Algorithmus, Fall 2: Ausgabe eindeutig

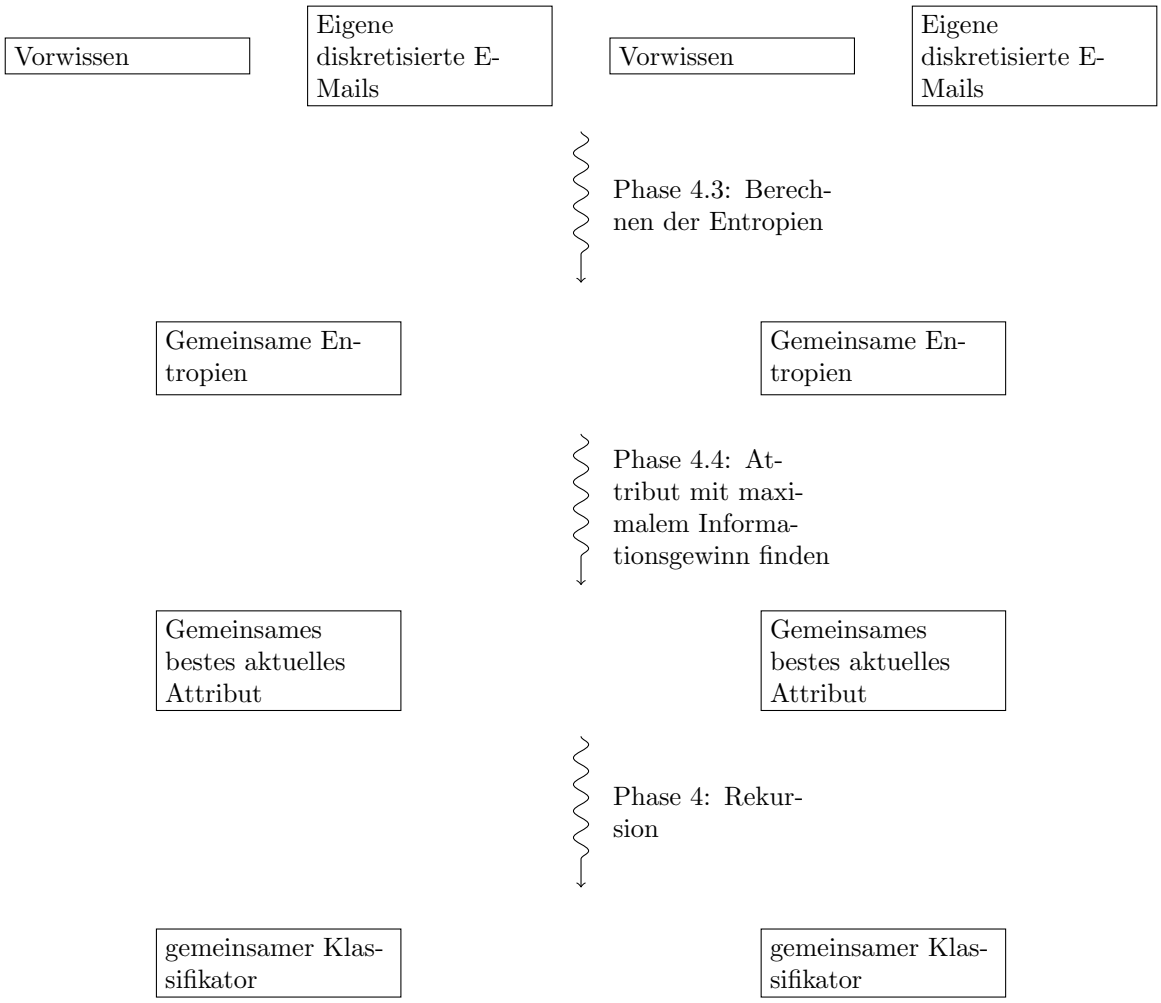


Figure 6: ID3-Algorithmus, Fall 3: Erzeugung eines Astes