Hochschule Konstanz Technik, Wirtschaft und Gestaltung (HTWG)
Fakultät Informatik
Rechner- und Kommunikationsnetze
Prof. Dr. Dirk Staehle

Labor zur Vorlesung Kommunikationstechnik

Laborübung zu Entropie

Prof. Dr. Dirk Staehle

Bearbeitung in Zweier-Teams

Team-Mitglied 1:

Team-Mitglied 2:

1 Informationsgehalt

In der Vorlesung haben wir gelernt, dass Information einer Reduktion der Anzahl von Möglichkeiten entspricht. Genaugenommen gilt für eine Reduktion von n auf m Möglichkeiten, dass wir eine Information von $I = \log_2 n/m$ gewonnen haben.

In der Vorlesung hatten wir für das Beispiel Xylophon festgestellt, dass das „X“ mehr Information bringt als das „n“ am Ende. In dieser Laboraufgabe wollen wir das genauer untersuchen. Die Aufgaben sollen alle so gelöst werden, dass Skripte implementiert werden, die für beliebige Wörter und nicht nur für Xylophon laufen.

Eine Liste (fast) aller deutschen Wörter finden Sie z.B. auf

https://github.com/davidak/wortliste/blob/master/wortliste.txt

1.1 Information der Wortlänge

Bestimmen Sie die Information, die wir gewinnen, wenn wir die Länge eines Wortes kennen. Welche Information gewinnen wir z.B. wenn wir wissen, dass ein Wort 5 Buchstaben hat?

Stellen Sie den Zusammenhang zwischen Wortlänge und gewonnener Information in einer Grafik dar.

X-Achse: Wortlänge

Y-Achse: Information

Tipps:

- from math import log2
- from collections import Counter, OrderedDict
- kompakt Listen und dictionaries erstellen:
    - bL=[a for word in aL]
    - bD={key,value for key,value in aD.items()}
- Erstellen Sie die Grafik mit
    - matplotlib (https://matplotlib.org/)
    - plotly (https://plotly.com/python/) bzw. plotly express
    - im Internet finden Sie zahlreiche Diskussionen, was (wofür) besser ist

1.2 Information der einzelnen Buchstaben

Bestimmen Sie für die einzelnen Buchstaben in dem Wort „Xylofon“, welche Information Sie beitragen, wenn Sie an „ihrer Position“ in einem Wort mit 8 Buchstaben stehen.

Also z.B. welche Information gewinnen wir, wenn wir wissen, dass in einem Wort mit 8 Buchstaben an Position 3 ein „l“ steht.

Stellen Sie das Ergebnis als Balkendiagramm dar. Verwenden Sie als „xticklabels“ die Buchstaben

1.3 Von vorne nach hinten und von hinten nach vorne

Wir versuchen jetzt ähnlich wie beim Orakelspiel in der Vorlesung zu bestimmen, welche Information wir gewinnen, wenn wir einen Buchstaben nach dem anderen vom Wort „Xylophon“ erraten. Dabei wollen wir einmal das Wort von vorne nach hinten und einmal von hinten nach vorne durchlaufen.

Bestimmen Sie jeweils, welche Information wir pro richtig geratenem Buchstaben gewinnen. Welche Information gewinnen wir, wenn wir wissen, dass ein Wort mit „X“ beginnt? Welche Information gewinnen wir, wenn wir wissen, dass ein Wort mit „n“ endet?

Stellen Sie den Informationsgewinn pro Buchstaben grafisch dar.

2 Nachrichtenquelle

In dieser Aufgabe soll eine Klasse Nachrichtenquelle (oder Source) implementiert werden. Der Konstruktor der Nachrichtenquelle erhält ein Wort, das die Auftrittswahrscheinlichkeiten der Zeichen für die Nachrichtenquelle festlegt. Bestimmen Sie im Konstruktor außerdem die Auftrittswahrscheinlichkeit der Zeichen, den Informationsgehalt der Zeichen sowie die Entropie der Zeichen. Speichern Sie diese Werte als Attribute.

Tipp: Es bietet sich hier an Dictionaries mit den Zeichen als Key zu verwenden

2.1 Test

Testen Sie ihre Klasse mit dem Wort Hochschule

2.2 RFC 2334

Im weiteren Verlauf der Aufgabe wollen wir den RFC 2334 „Hyper Text Coffee Pot Control Protocol“ (https://www.rfc-editor.org/rfc/rfc2324, auch Standardisierer versuchen lustig zu sein) als Beispiel verwenden. Generieren Sie eine Nachrichtenquelle nach RFC 2334.

Stellen Sie die Häufigkeit der Zeichen und den Informationsgehalt der Zeichen graphisch dar:

X-Achse: Zeichen (geordnet nach absteigender Auftrittswahrscheinlichkeit)

Y-Achse: Häufigkeit/Informationsgehalt der Zeichen

3 Entropiecodierung

In dieser Aufgabe geht es um die Entropiecodierung. In Aufgabe 3.1 sollen Sie entweder die Codierungsvorschrift nach Shannon oder nach Hoffman implementieren. In Aufgabe 3.2 soll eine Klasse Code erstellt werden, die die Codierung bzw. Dekodierung nach der Codierungsvorschrift durchführt. In Aufgabe 3.3 sollen dann Shannon, Huffman und arithmetische Codierung vergleichen werden.

3.1 Codierungsvorschrift

Implementieren Sie eine Funktion zur Erstellung der Codierungsvorschrift nach Shannon oder nach Huffman. Das entspricht den Beispielen, die wir in der Vorlesung durchgeführt haben. Finden Sie sich mit einem Partner-Team zusammen, so dass ein Team Shannon und eine Team Huffman implementiert. Danach können Sie den Code austauschen und gegenseitig testen.

Die Codierungsvorschrift soll das Dictionary implementiert werden, das einen Buchstaben als „Key“ auf den Binär-Code als „Value“ abbildet.

Tipps zu Shannon:

- verwenden Sie als Datenstruktur eine Liste mit Tupeln aus Häufigkeit, Zeichen und Codierung pro Zeichen
- implementieren Sie die Codierung rekursiv, in dem Sie die Codierung jeweils für die obere und untere Hälfte aufrufen
- erweitern Sie die Codierung in jedem Rekursionsschritt

Tipps zu Huffman:

- Verwenden Sie als Datenstruktur eine sortierte Liste. Hier bietet sich die Funktionen heapify, heappop und heappush aus dem Modul heapq an

3.1.1 Test

Testen Sie die Codierungsvorschriften mit den Beispielen aus der Vorlesung.

3.2 Codieren und dekodieren

In dieser Aufgabe soll eine Klasse Code zum Codieren und Dekodieren von Nachrichten entsprechend einer Codierungsvorschrift implementiert werden. Der Konstruktor der Klasse enthält eine Nachrichtenquelle sowie eine Funktion zum Erstellen der Codierungsvorschrift, d.h. entweder shannon oder huffman.

Im Konstruktor wird die Codierungsvorschrift erstellt und zusätzlich die mittlere Codewortlänge sowie die Redundanz des Codes bestimmt. Speichern Sie die Nachrichtenquelle sowie die berechneten Werte als Attribute.

Weiterhin enthält die Klasse die Methoden

- code, av_char_length, redundancy=encode(word)
- word=decode(code)

Die Methode encode liefert neben dem Binär-Code auch noch die mittlere Anzahl Bits pro Zeichen sowie die daraus resultierende „Redundanz“[1] zurück. Beachten Sie, dass diese Werte nicht unbedingt der mittleren Codewortlänge und Redundanz der Nachrichtenquelle entsprechen, sondern davon abweichen können, da die Häufigkeiten der Zeichen in der codierten Nachricht nicht exakt den Häufigkeiten der Nachrichtenquelle entsprechen.

3.2.1 Test

Testen Sie die Codierung und Dekodierung mit den Beispielen aus der Vorlesung. Wählen Sie auch eine auf „Hochschule“ basierende Nachrichtenquelle und codieren Sie die Nachrichten „HHHHH“ bzw. „EEEEE“. Vergleichen Sie die mittlere Anzahl Bits pro Zeichen und die „Redundanzen“.

---

[1] Der Begriff Redundanz ist hier eigentlich nicht korrekt, weil sich die Redundanz immer auf die mittlere Codewortlänge bezieht.

4 Vergleich

In dieser Aufgabe wollen wir die Codierungen nach Shannon, Huffman und die arithmetische Codierung vergleichen.

4.1 Arithmetische Codierung

Verwenden Sie für die arithmetische Codierung die Arithmetic Coding Library https://pypi.org/project/arithmetic-compressor/ mit dem StaticModel. Das StaticModel ist auf eine feste minimale Wahrscheinlichkeit festgelegt, die sie leider nur im Code selbst ändern können. Öffnen Sie daher das StaticModel und ändern Sie den SCALE_FACTOR auf 2**16. Überlegen Sie sich daher, ob Sie das Package über pip oder lokal installieren. Codierung und Dekodierung der Daten erfolgt dann wie auf in der Library beschrieben.

4.2 RFC 2334

Vergleichen Sie die Qualität der Codierung anhand des RFC 2334.

Bemerkung: Hier entspricht die Auftrittswahrscheinlichkeit der Zeichen in der Nachricht exakt der vorgegebenen Auftrittswahrscheinlichkeiten, nach denen die Codierungsvorschrift erstellt wurde.

4.3 Zufällige Nachrichten aus RFC 2334

Im zweiten Schritt wählen wir 1000 zufällige Zeichen aus dem RFC 2334 und führen die Codierung mit den drei Verfahren durch. Das wiederholen wir zehnmal.

Stellen Sie die resultierenden Codierungslängen als Balkendiagram dar.

Hinweis: Zufällige Zeichen können Sie einfach mit der Funktion choices auf dem Modul random erzeugen.