Beleg 1

Ziel des ersten Belegs ist es, funktionales Programmieren mit rekursiven Datenstrukturen und das Anwenden von Higher Order Functions zu üben. Das Beleg-Projekt befindet sich im File Beleg1ListsAufgabe.zip. Entpacken Sie das Projekt und öffnen Sie es mit Ihrer Entwicklungsumgebung. Das Projekt teilt sich in 2 Aufgabenbereiche: Erstens die Implementierung von Funktionen einer Liste und zweitens die Anwendung von Higher Order Functions in einer Sentiment-Analyse. Verwenden Sie für die Implementierung nur Elemente der Funktionalen Programmierung also z.B. keine Variablen. Für alle Funktionen finden Sie Tests in den entsprechenden Unterverzeichnissen. Diese dienen nicht nur der Überprüfung der Funktion sondern auch als Spezifikation der Funktionen. Haben Sie also Fragen zu den einzelnen Funktionen, können Sie auch erst einmal in die Tests hinein schauen und überlegen, was die Funktionen genau liefern sollen.

Teil 1

Aufgabe 1: Starten Sie mit der Programmierung der Listenfunktionen. Die zu implementierenden Funktionen finden Sie in der Klasse SinglyLinkedIntList – eine Beschreibung der Funktionen im Interface (trait) IntList.

Teil 2

Ausgangspunkt des zweiten Teils sind Textdateien, für die verschiedenste Statistiken erstellt werden sollen. Weiter sollen die Dateien effizient durchsuchbar gemacht werden. Das Grundgerüst für alle zu schreibenden Funktionen sowie verschiedene Tests finden Sie in den Klassen Sentiments, Processing, SentimentTest sowie ProcessingTest.

Weiter befinden sich sich im Testressourcen-Ordner verschiedene Beispieldateien, die für die Analyse verwendet werden können/sollen: MobyDick.txt (Buch Moby Dick von Herman Melville), MobyDickC1.txt (Kapitel 1 von Moby Dick), MobyDickShort.txt (Teil aus dem ersten Kapitel), Robinson.txt (Buch “Robinson Crusoe” von Daniel Defoe) sowie AFINN-111.txt (Beispieldatei für die Sentimentanalyse).

Aufgabe 1: Vervollständigen Sie die Funktionen getWords, getAllWords und countWords, die das Zählen von Wörtern innerhalb eines Texts ermöglichen sollen. Der Text befindet sich dabei im folgenden Format: Jede Zeile besteht aus einem Tupel (Int, String), das eine Zeilennummer enthält sowie die Textzeile als String. Beim Zählen der Wörter soll keine Unterscheidung zwischen Groß- und Kleinschreibung gemacht werden, d.h. Baum wäre bspw. das selbe Wort wie bauM. Gehen Sie dabei folgendermaßen vor: Implementieren Sie im ersten Schritt die Funktion getWords, die aus einem String alle Wörter extrahiert. Ersetzen Sie zu diesem Zweck zuerst alle Elemente, die keine Buchstaben sind, durch das Leerzeichen und splitten Sie dann den String auf Basis der Leerzeichen (Hinweis: Schauen Sie sich die Funktionen split und replaceAll dafür an)

Implementieren Sie danach die Funktion getAllWords, die aus dem Gesamttext (Liste von Tupeln (Zeilennr, Text) alle Wörter extrahiert. Sind die ersten beiden Funktionen implementiert, so kann eine Liste aller im Text vorkommenden Wörter generiert werden, welche wiederum Input der Funktion countWords ist. Sie zählt alle Vorkommen eines Wortes und gibt eine Liste bestehend aus den Wörtern und deren Anzahl zurück. Verwenden Sie dafür die Datenstruktur Map.

Aufgabe 2: In dieser Aufgabe sollen große Texte durchsuchbar gemacht werden, d.h. es soll nach Zeilen gesucht werden können, in denen beliebig gewählte Schlüsselwörter vorkommen. Dazu soll im ersten Schritt ein sogenannter Inverser Index erstellt werden. Im Invesen Index soll gespeichert werden, in welchen Zeilen innerhalb eines Dokuments ein Wort vorkommt. Implementieren Sie hier zunächst die Funktion getAllWordsWithIndex. Diese Funktion bekommt den Text im Format List((Zeilennr, Zeilentext)), extrahiert alle Wörter und speichert diese als Tupel in der Form (Zeilennr, Wort). Jetzt müssen in der Funktion createInverseIndex sämtliche Vorkommen eines Wortes (Zeilennummern) innerhalb einer Map zusammengefasst werden. Ergebnis der Funktion ist somit eine Map,die ein Wort auf eine Liste von Zeilennummern (Int) abbildet. Auf dieser Basis können jetzt die Funktionen andConjunction und orConjunction implementiert werden. Ziel der Funktionen ist es, für eine Liste von Wörtern die Zeilen herauszusuchen, die diese enthalten. Dabei sollen bei orConjunction die Wörter “verodert” werden und die Zeilen zurückgegeben werden, bei denen mindestens eins der Wörter vorkommt. Bei der andConjunction sollen die Wörter mit dem logischen und verknüpft werden und nur Zeilen zurückgegeben werden, in denen alle Wörter vorkommen.

Aufgabe 3: In dieser Aufgabe soll eine Sentiment-Analyse durchgeführt werden. In einer Sentiment-Analyse werden gefühlsausdrückenden Wörtern ein Wert von +5 (positiv) bis -5 (negativ) zugeordnet. Diese Zuordnung befindet sich in der Datei AFINN-111.txt. In der vorgegebenen Funktion getSentimentswird diese Datei ausgelesen und im Format einer Map [String, Int] bereitgestellt. Auf Basis dieser Sentiment-Zuordnung soll eine Analyse durchgeführt werden. Implementieren Sie hierzu als erstes die Funktion getDocumentGroupedByCounts. Diese bekommt als Parameter einen Dateinamen sowie die Anzahl von Wörtern, die innerhalb eines Abschnitts zusammengefasst werden sollen. Ergebnis der Funktion ist dann ein Tupel,bestehend aus der Abschnittsnummer und einer Liste von Wörtern, die dort vorkommen. Die Abschnittsnummern sollten bei 1 anfangen.

Implementieren sie jetzt eine Funktion getDocumentSplitByPredicate. Diese ist ähnlich zu getDocumentGroupedByCounts und bekommt als Parameter einen Dateinamen sowie ein Predikat (Funktion mit booleschem Rückgabewert). Es soll dazu benutzt werden, die Abschnitte zu unterteilen. Jedes Mal wenn die Funktion für eine Zeile „true“ zurückgibt, soll ein neuer Abschnitt begonnen werden. Der Text, der vor dem ersten Abschnitt steht, soll dabei verworfen werden.

Letzte zu implementierende Funktion ist analyseSentiments. Diese bekommt eine Liste von Abschnitten (Abschnittnr, Liste von Wörtern) und errechnet den jeweiligen Sentimentwert. Der Sentimentwert ist der Durchschnitt der Sentimentwerte aller bekannten Wörtern des betrachteten Abschnitts. Ergebnis ist ein Tripel, bestehend aus der Abschnittsnummer, dem Sentimentwert und der relativen Anzahl von Wörtern, die für die Sentimentanalyse verwendet werden konnten. In der Klasse App befindet sich eine Mainklasse, die für das Buch „Robinson Crusoe“ die Ergebnisse der Sentimentanalyse graphisch darstellt. Testen. Sie diese und überprüfen Sie die Plausibilität der Ergebnisse.

****

Der Beleg soll in Gruppen mit 2 Personen gelöst und bis zum 01.06.2023 (23:59 Uhr) abgegeben werden. Dabei muss jeder Teilnehmer des Kurses seine Lösung abgeben und kenntlich machen, mit wem er zusammen gearbeitet hat. Die Abnahme des Belegs erfolgt in den Übungen am 05.06.2023. Dabei müssen beide Gruppenteilnehmer anwesend sein – und auch beide die Aufgaben des Belegs erklären können müssen.

Bei Abgabe müssen alle Aufgaben gelöst worden sein. Probleme und Fragen können in den vorigen Übungen oder den eingerichteten Foren geklärt werden.