LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
CENTRUM FÜR INFORMATIONS- UND SPRACHVERARBEITUNG
STUDIENGANG COMPUTERLINGUISTIK

NACHHOLKLAUSUR ZUR ÜBUNG BACHELORMODUL ,,SYMBOLISCHE PROGRAMMIERSPRACHE“ WS 2020/2021

FLORIAN FINK

NACHHOLKLAUSUR ZUR ÜBUNG AM 13.04.2021

VORNAME:	
NACHNAME:	
MATRIKELNUMMER:	
STUDIENGANG:	B.Sc. Computerlinguistik, B.Sc. Informatik, Magister

Die Nachholklausur zur Übung besteht aus **6 Aufgaben** und umfasst **11 Seiten**. Die Punktzahl ist bei jeder Aufgabe angegeben. Die Bearbeitungsdauer beträgt **45 Minuten**. Bitte überprüfen Sie, ob Sie ein vollständiges Exemplar erhalten haben.

Aufgabe	mögliche Punkte	erreichte Punkte
1. Evaluierung von Klassifikatoren	4	
2. Quiz	3	
3. Git	8	
4. Dokumenten-Retrieval und Unit-Tests	5	
5. K-Means	6	
6. POS Tagging	5	
Summe	31	
Note		

Einwilligungserklärung
Hiermit stimme ich einer Veröffentlichung meines Klausurergebnisses unter Verwendung meiner Matrikelnummer im Internet zu.

Datum: _____ Unterschrift: _____

NAME:

Aufgabe 1 *Evaluierung von Klassifikatoren* [4 Punkte]
Gegeben seien ein Klassifikator für die Klassen *A*, *B* und *C* und zwei Listen, die jeweils die vorhergesagten bzw. tatsächlichen Klassen (*A*, *B* und *C*) einer Testmenge enthalten.

(a) Implementieren Sie die Funktion unten, die den Recall für eine angegebene Klasse berechnen soll.

```
# Beispielargumente
# example_y = ["A", "C", "B", "C", "A"]
# example_pred = ["B", "C", "A", "C", "B"]
# example_c = "B"
def recall(y, pred, c)
```

(b) Implementieren Sie die Funktion unten, die die Accuracy berechnen soll.

```
# Beispielargumente
# example_y = ["A", "C", "B", "C", "A"]
# example_pred = ["B", "C", "A", "C", "B"]
def accuracy(y, pred)
```

(2+2 = 4 Punkte)

NAME:

NAME:

Aufgabe 2 *Quiz* [3 Punkte]

Beantworten Sie folgende Fragen.

(a)
```
x = [p for p in range(1, 11) if p%2 == 0]
```
Was enthält die Variable x?

(b)
```
import math
def x(y):
    return math.sqrt(sum(x*x for x in y))
print(x([1, 2, 3]))
```
Was wird in der Funktion x berechnet?

(c)
```
x = {a: b for a, b in zip(["a", "b"], [1, 2])}
```
Was enthält die Variable x?

(1+1+1 = 3 Punkte)

NAME:

Aufgabe 3 *Git* [8 Punkte]

Sie arbeiten mit mehreren Teammitgliedern an einer Aufgabe und verwenden `git` (mit Gitlab remote) als Versionskontrolle.

(a) Erklären Sie kurz die Funktion der Befehle `git clone https://myurl.com/myrepo`, `git log`, `git checkout mybranch` und `git status`.

(b) Beschreiben Sie das Vorgehen, um die aktuellste Version vom Server zu holen, eigene Änderungen in eine Datei einzubringen und Ihre Änderungen auf den Server hochzuladen. Der Remote-Name sei dabei `origin`, der Branch-Name `master` und die Datei, die Sie verändern, heiße `file.py`.

(4 + 4 = 8 Punkte)

NAME:

Aufgabe 4 *Dokumenten-Retrieval und Unit-Tests* [5 Punkte]
Gegeben seien folgende Implementierungen einer Dokumentenkolletkion und einer Dokumentenklasse:

```
import nltk
class Document:
    def __init__(self, text):
        self.text = text
        self.counts = nltk.FreqDist(nltk.word_tokenize(text))
```

(a) Implementieren Sie die Funktion `cosinus`, die den Kosinus zwischen zwei Dokumentenvektoren berechnet.

```
# Beispiel:
# example_d1=Document("first document")
# example_d2=Document("second document")
# vec = consinus(example_d2, example_d1)
def consinus(d1, d2)
```

(b) Implementieren Sie eine Unit-Testklasse, in der Sie die `cosinus` Funktion aus Aufgabe (a) testen.

(3+2 = 5 Punkte)

NAME:

NAME:

Aufgabe 5 *K-Means* [6 Punkte]
Implementieren Sie folgende Funktionen.

(a) Die Funktion `distance` soll den Euklidischen Abstand zwischen zwei Vektoren berechnen. Die Vektoren sind dabei Dictionaries, die Wörter auf ihre Gewichte abbilden.

```
# Beispiel:
# example_v1 = {'a': 2, 'b': 3}
# example_v2 = {'b': 1, 'c': 1}
# distance(example_v1, example_v2) # result is 3.0
def distance(v1, v2)
```

(b) Die Funktion `cluster` soll den Index des Clusterzentroiden zurückliefern, der am nächsten zu dem Vektor ist. Vektoren sind dabei Dictionaries, die Wörter auf ihre Gewichte abbilden. Die Clusterzentroiden sind in einer Liste von Vektoren abgelegt. Sie können die Funktion `distance` aus der vorherigen Aufgabe verwenden.

```
# Beispiel:
# example_means=[{'a':1}, {'b': 2}, {'c': 3}]
# example_v={'a': 2, 'c': 3}
# cluster(example_means, example_v) # result is 1
def cluster(means, v)
```

(3+3 = 6 Punkte)

NAME:

NAME:

Aufgabe 6 *POS Tagging* [5 Punkte]

Implementieren Sie ein Programm mit NLTK, mit dem man die häufigsten 3 POS-Tags in einem Korpus bestimmen kann. Vergessen Sie auch die entsprechenden `import`-Ausdrücke nicht.

NAME: