LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

CENTRUM FÜR INFORMATIONS- UND SPRACHVERARBEITUNG
STUDIENGANG COMPUTERLINGUISTIK

PROBEKLAUSUR ZUM BACHELORMODUL ,,SYMBOLISCHE PROGRAMMIERSPRACHE“ WS 2020/2021

FLORIAN FINK

KLAUSUR AM 22.12.2020

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

Die Klausur besteht aus **8 Aufgaben**. Die Punktzahl ist bei jeder Aufgabe angegeben. Die Bearbeitungsdauer beträgt **45 Minuten**. Bitte überprüfen Sie, ob Sie ein vollständiges Exemplar erhalten haben.
Die ersten 4 Aufgaben spiegeln Aufgaben für die Vorlesung, die letzten 4 Aufgaben spiegeln Aufgaben für die Übung wieder.

Aufgabe	mögliche Punkte	erreichte Punkte
1. Precision, Recall und Accuracy	6	
2. Berichtigung falscher Aussagen	6	
3. KNN	5	
4. Python	6	
5. Precision, Recall und F1-Measure	6	
6. Unit-Testing	4	
7. Git	6	
8. Naive Bayes Klassifikator	6	
Summe	45	
Note		

NAME:

Aufgabe 1 *Precision, Recall und Accuracy* [6 Punkte]
Ein Klassifizierer teilt Texte in die Klassen A und B ein. Bei der Auswertung ergibt sich folgende Tabelle:

		automatisch zugewiesene Klasse	
		A	B
tatsächliche Klasse	A	43	17
	B	15	25

Berechnen Sie

(a) die Accuracy des Klassifizierers,

(b) die Precision des Klassifizierers für Klasse B,

(c) den Recall des Klassifizierers für Klasse B,

Ihr Rechenweg muss nachvollziehbar sein.

(2+2+2 = 6 Punkte)

NAME:

Aufgabe 2 *Stellen Sie folgende falschen Aussagen richtig* [6 Punkte]

(a) Der K-Means Algorithmus ist ein überwachter (supervised) Algorithmus zur Dokumentenklassifikation.

(b) Die bedingte Wahrscheinlichkeit $P(A|B)$ gibt die Wahrscheinlichkeit für B an, falls A eingetreten ist.

(c) Die Wahrscheinlichkeit $P(\neg A)$ (Ereignis A tritt *nicht* ein) ist $\frac{1}{P(A)}$.

(d) Für die Wahrscheinlichkeit $P(A)$ eines Ereignisses A gilt: $0 \leq P(A) \leq 100$.

(e) Die Länge $|\vec{x}|$ des Vektors $\vec{x} = (4, 3)$ ist $\sqrt{7}$.

(f) Das Skalarprodukt $\vec{x} \cdot \vec{y}$ zwischen den Vektoren $\vec{x} = (1, 2, 3)$ und $\vec{y} = (2, 2, 2)$ ist der Vektor $(2, 4, 6)$.

(1+1+1+1+1+1 = 6 Punkte)

NAME:

Aufgabe 3 *KNN* [5 Punkte]
Beschreiben Sie den K-nearest-neighbour Algorithmus zur Klassifikation von Dokumenten. Gehen Sie dabei auch auf die Dokumentenrepresentation (Features), das Ähnlichkeitsmaß und die abschließende Klassifikation ein.

NAME:

Aufgabe 4 *Python* [6 Punkte]

(a) Warum scheitert folgender Python-Code mit einem Fehler? Wie kann man den Fehler beheben?

```
>>> class Person:
...     def __str__(self):
...         return self.first_name + ' ' + self.last_name
...
>>> p = Person()
>>> print(p)
```

(b) Was ist das Attribut x der Klasse X im folgende Python-Code? Was berechnet die Methode y der Klasse X?

```
>>> from collections import Counter
>>> class X:
...    def __init__(self, xs):
...         self.x = Counter(z)
...    def y(self, z):
...        return (self.x.get(z,0)+1) / (len(self.x.keys())+sum(self.x.values()))
...
>>> x = X(['a', 'b', 'c', 'a'])
>>> print(x.y('d'))
```

NAME:

(c) Was berechnen die beiden Funktionen y und z?

```
>>> from collections import Counter
>>> from math import sqrt
>>> class X:
...     def __init__(self, xs):
...         self.x = Counter(z)
...
>>> def y(a, b):
...    return sum([a[c] * b.get(c, 0) for c in a])
...
>>> def z(a, b):
...    return (y(a, b)) / (sqrt(y(a, a)) * sqrt(y(b, b)))
...
>>> x1 = X(['a', 'b', 'c', 'a'])
>>> x2 = X(['a', 'b', 'c'])
>>> print(z(x1.x, x2.x))
```

(2+2+2 = 6 Punkte)

NAME:

Aufgabe 5 *Precision, Recall und F1-Measure* [6 Punkte]
Gegeben ein binärer Klassifikator für die Klassen *True* und *False*.

(a) Gegeben zwei Listen, die jeweils die vorhergesagten bzw. tatsächlichen Labels (*True* und *False*) einer Testmenge enthalten. Verfolständigen Sie die Funktion unten, die die Precision für die Klasse `True` berechnen soll.

```
# Beispielargumente
example_y = [True, False, False, True]
example_pred = [True, True, True, False]
def precision(y, pred):
```

NAME:

(b) Gegeben zwei Listen, die jeweils die vorhergesagten bzw. tatsächlichen Labels (*True* und *False*) einer Testmenge enthalten. Verfolständigen Sie die Funktion unten, die den Recall für die Klasse `True` berechnen soll.

```
# Beispielargumente
example_y = [True, False, False, True]
example_pred = [True, True, True, False]
def recall(y, pred):
```

NAME:

(c) Gegeben zwei Listen, die jeweils die vorhergesagten bzw. tatsächlichen Labels (*True* und *False*) einer Testmenge enthalten. Verfolständigen Sie die Funktion unten, die das F1-Measure für die Klasse `True` berechnen soll (sie können die Funktionen aus den vorherigen beiden Aufgaben (a) und (b) verwenden).

```
# Beispielargumente
example_y = [True, False, False, True]
example_pred = [True, True, True, False]
def f1(y, pred):
```

(2+2+2 = 6 Punkte)

NAME:

Aufgabe 6 *Unit-Testing* [4 Punkte]

Was ist der Unterschied zwischen dem `doctest` und dem `unittest` Modul? Definieren Sie eine Funktion `my_square(x)`, die Zahlen quadriert und Schreiben Sie dafür je einen Test mit `doctest` und einen `unittest`.

NAME:

Aufgabe 7 *Git* [6 Punkte]

Sie arbeiten in einem Git-Repository auf dem `work` Zweig (branch). Gerade haben Sie die Dateien `a.py` und `b.py` bearbeitet. Dies ist die Ausgabe von `git status`:

```
$ git status
On branch work
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)
modified:   a.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)
b.py
no changes added to commit (use "git add" and/or "git commit -a")
```

Geben Sie die `git`-Befehle in der richtigen Reihenfolge an um

(a) die Änderungen in den beiden Dateien `a.py` und `b.py` in das Repository auf dem `work` Zweig einzubringen (vergessen Sie die Commit-Nachricht nicht),

(b) Ihre Änderungen im `work` Zweig in den `master` Zweig zu mergen (vergessen Sie dabei nicht, dass Sie sich auf dem `work` Zweig befinden),

(c) Ihre Änderungen im `master` Zweig an das Remote-Repository `origin` zu schicken (vergessen Sie dabei nicht, vor dem Hochladen mögliche Änderungen von dem Remote-Repositiory herunter zu laden).

(2+2+2 = 6 Punkte)

NAME:

Aufgabe 8 *Naive Bayes Klassifikator* [6 Punkte]

(a) Vervollständigen Sie die Funktion `log_probability`, die die logarithmierte Wahrscheinlichkeit $\log P(word)$ eines Wortes aus der Anzahl der Vorkommen des Wortes im Korpus `wordcount` (Integer), der Vokabulargröße `vocab_size` (Integer) und der Summe aller Wortvorkommen `total` (Integer) berechnet. Verwenden Sie addiere-λ-Glättung, mit dem Parameter `smoothing` (Float).

```
def log_probability(wordcount, vocab_size, total, smoothing):
```

(b) Vervollständigen Sie die Funktion `sentence_log_probability`, die die logarithmierte Wahrscheinlichkeit $\log P(sentence)$ eines Satzes `sentence` (Liste von Strings) berechnet (nehmen Sie dabei statistische Unabhängigkeit der Wörter an im Satz an). Das Dictionary `word_to_count` hält alle Wörter des Vokabulars als Schlüssel (keys) und die Anzahl der entsprechenden Wortvorkommen im Korpus als Werte (values). Wie oben gibt `smoothing` (Float) den Glättungsparameter an. Sie können `log_probability` von Teil (a) der Aufgabe verwenden.

```
def sentence_log_probability(sentence, word_to_count, smoothing):
```

(3+3 = 6 Punkte)

NAME:

```
>>> # Schöne Feiertage.
>>> for x in range(1, 30, 2):
...     print(('*' * x).center(30))
```