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

KLAUSUR ZUR VORLESUNG BACHELORMODUL ,,SYMBOLISCHE PROGRAMMIERSPRACHE“ WS 2020/2021

FLORIAN FINK

KLAUSUR ZUR VORLESUNG AM 18.02.2021

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

Die Klausur zur Vorlesung besteht aus **7 Aufgaben** und umfasst **10 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. Precision, Recall und Accuracy	6	
2. Quizz	6	
3. K-means	7	
4. Objektorientierung	3	
5. TF-IDF	6	
6. NLTK und lexikalische Information	7	
7. WordNet	5	
Summe	40	
Note		

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

Datum: _____ Unterschrift: _____

NAME:

Aufgabe 1 *Precision, Recall und Accuracy* [6 Punkte]
Ein Klassifizierer teilt 100 Texte in die 4 Klassen A, B, C und D ein. Bei der Auswertung ergibt sich folgende Wahrheitsmatrix (confusion matrix):

```
  |  A  B  C  D |
--+-------------+
A |<30> 7  13 3 |
B |  8 <1> 4  3 |
C |  7  5 <3> 3 |
D |  7  1  2 <3>|
--+-------------+
(row = reference; col = test)
```

Berechnen Sie

(a) die Accuracy des Klassifizierers,

(b) die Precision des Klassifizierers für Klasse A,

(c) den Recall des Klassifizierers für Klasse A,

Ihr Rechenweg muss nachvollziehbar sein.

(2+2+2 = 6 Punkte)

NAME:

Aufgabe 2 *Quizz* [6 Punkte]
Bearbeiten Sie folgende Aufgaben.

(a) Geben Sie einen Unterschied zwischen überwachten (supervised) und unüberwachten (unsupervised) Algorithmen an.

(b) Geben Sie das Entscheidungskriterium bei der binären Klassifikation (Naive-Bayes) an, um `text` in die Klassen `A` oder `B` zu klassifizieren.

(c) Geben Sie die Wahrscheinlichkeit von $P(\neg A)$ (Ereignis A tritt *nicht* ein) an.

(d) (Anmerkung: Das Vorkommen von Worten ist im Allgemeinen *nicht statistisch unabhängig*). Geben Sie an, wie die Wahrscheinlichkeit des Satzes *„heute regnet es"* berechnet wird (Wahrscheinlichkeit von Wortsequenzen).

(e) Berechnen Sie die Länge $|\vec{x}|$ des Vektors $\vec{x} = (4, 3)$.

(f) Berechnen Sie das Skalarprodukt $\vec{x} \cdot \vec{y}$ zwischen den Vektoren $\vec{x} = (1, 2, 3)$ und $\vec{y} = (3, 2, 3)$.

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

NAME:

Aufgabe 3 *K-means* [7 Punkte]

Der K-means Algorithmus ist ein unüberwachter (unsupervised) Algorithmus zur Klassifikation.

(a) Beschreiben Sie den K-means Algorithmus zur Klassifikation von Dokumenten. Gehen Sie dabei auch auf die Dokumentenrepresentation (Features), das Ähnlichkeitsmaß, die Berechnung der Cluster-Zentroide und die Klassifizierung ein.

(b) Inwieweit handelt es sich bei K-means um einen *unüberwachten* (unsupervised) Algorithmus?

(5+2 = 7 Punkte)

NAME:

Aufgabe 4 *Objektorientierung* [3 Punkte]

```
import nltk
import os

class DocumentCollection:
    def __init__(self, docs):
        self.docs = docs
    @classmethod
    def x(cls, d):
        cls([Document.from_file(p) for p in os.listdir(d) if p.endswith(".txt")])
    def y(self, term):
        docs = [d for d in self.docs if term in d.counts]
        return len(docs)

class Document:
    def __init__(self, counts):
        self.counts = counts
    @classmethod
    def from_file(cls, path):
        with open(path, mode='r', encoding='utf-8') as f:
            return cls(nltk.FreqDist(nltk.word_tokenize(f.read())))
    def term_frequency(self, term):
        return self.counts.get(term, 0)
```

Betrachten Sie folgenden objektorientierten Python-Code.

(a) Was ist die Bedeutung der `self` Variable in den beiden Klassen?

(b) Was genau macht die Funktion `x` in der `DocumentCollection` Klasse?

(c) Was genau berechnet die Methode `y` in der `DocumentCollection` Klasse?

(1+1+1 = 3 Punkte)

NAME:

NAME:

Aufgabe 5 *TF-IDF* [6 Punkte]

Gegeben sei die Formel

$$tf.idf_{d,t} = tf_{d,t} \log_{10} \frac{N}{df_t}$$

zur Berechnung der TF-IDF-Gewichte von Termen in Dokumenten.

(a) Erläutern Sie die Variablen $tf_{d,t}$, N und df_t in der Formel.

(b) Erläutern Sie inwieweit diese TF-IDF Gewichtung die Relevanz von Termen in Dokumenten für die Dokumentensuche berechnet (welche Terme haben ein hohes Gewicht, welche haben ein geringes Gewicht und inwieweit unterstützten diese Gewichtungen die Dokumentensuche).

(3+3 = 6 Punkte)

NAME:

Aufgabe 6 *NLTK und lexikalische Information* [7 Punkte]

(a) Definieren Sie die Begriffe *Token*, *Type* und *Bigram*.

(b) Nennen und erläutern Sie zwei Möglichkeiten die Corpora in NLTK für computerlinguistische Anwendungen zu verwenden.

(b) Was enthält die Variable `result` in der letzten Zeile des folgenden Code-Ausschnitts?

```
import nltk

words = nltk.corpus.brown.words(categories="fiction")
bigrams = nltk.bigrams(words)
cfd = nltk.ConditionalFreqDist(bigrams)

print (list(cfd["living"].values()))
>>> [8, 2, 1, 1, 2, 1, 1, 1]

print(list(cfd["living"]))
>>> ['room', 'together', 'in', 'under', 'symbol', 'things', 'of', 'a']

result = cfd["living"].max()
```

(3+2+2 = 7 Punkte)

NAME:

NAME:

Aufgabe 7 *WordNet* [5 Punkte]

(a) Beschreiben Sie den Lesk-Algorithmus.

(b) Skizzieren Sie wie man den Lesk-Algorithmus im NLTK unter Verwendung von WordNet implementieren kann, um mehrdeutige Begriffe zu disambiguieren.

(2+3 = 5 Punkte)