LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

CENTRUM FÜR INFORMATIONS- UND SPRACHVERARBEITUNG
STUDIENGANG COMPUTERLINGUISTIK

KLAUSUR ZUM BACHELORMODUL „PROBEKLAUSUR ÜBUNG COMPUTERLINGUISTISCHE ANWENDUNGEN“

PROBEKLAUSUR,
DR. BENJAMIN ROTH
KLAUSUR AM

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

Die Klausur besteht aus **7 Aufgaben**. Die Punktzahl ist bei jeder Aufgabe angegeben. Die Bearbeitungsdauer beträgt **45 Minuten**. Bitte überprüfen Sie, ob Sie ein vollständiges Exemplar erhalten haben. Tragen Sie die Lösungen in den dafür vorgesehenen Raum im Anschluss an jede Aufgabe ein. Falls der Platz für Ihre Lösung nicht ausreicht, benutzen Sie bitte **nur** die ausgeteilten Zusatzblätter! Verwenden Sie einen dokumentenechten Kugelschreiber oder Füller, **keine** Bleistifte. Es sind **keine Hilfsmittel** zugelassen, außer ein **selbst von Hand beschriebenes DIN A4 - Blatt**. Geben Sie Programmcode immer in **Python** an. **Sie können Fragen auf Englisch bearbeiten**. Bitte tragen Sie **zuerst**, d.h., bevor Sie die Aufgaben lösen, auf **allen** Seiten Ihren Namen ein und füllen Sie die Titelseite aus.

Aufgabe	mögliche Punkte	erreichte Punkte
1. Evaluierung von Klassifikatoren	2	
2. Unit-Testing	4	
3. Dokumenten-Retrieval	6	
4. Git	4	
5. Naive Bayes Klassifikator	5	
6. K-nearest neighbors	4	
7. K-Means	4	
Summe	29	
Note		

Einwilligungserklärung (optional)

Hiermit stimme ich einer Veröffentlichung meines Klausurergebnisses in der Veranstaltung „PROBEKLAUSUR Übung Computerlinguistische Anwendungen“ vom unter Verwendung meiner Matrikelnummer im Internet zu.

Datum: _____ Unterschrift: _____

NAME:

Aufgabe 1 Evaluierung von Klassifikatoren

Gegeben ein binärer Klassifikator für die Klassen `True` und `False`.

(a) Gegeben zwei Listen, die jeweils die vorhergesagten bzw. tatsächlichen Labels (`True` bzw `False`) eines Testsets enthalten. Vervollständigen Sie die Funktion unten, die die Accuracy berechnen soll.

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



    return
```

2 PUNKTE

NAME:

Aufgabe 2 Unit-Testing

Was ist der Unterschied zwischen dem `doctest` und `unittest` Modul? Definieren Sie eine Funktion `my_square(x)`, die Zahlen quadriert, und schreiben Sie dafür je einen Test mit `doctest` und `unittest`. 4 PUNKTE

NAME:

Aufgabe 3 Dokumenten-Retrieval

Im Folgenden werden Bag-of-Words Dokumentvektoren wie in der Vorlesung als Dictionaries (Word → Count) repräsentiert.

(a) Vervollständigen Sie den Programmcode zur Berechnung des Vektor-Produkts (*dot product*) zweier Dokumentvektoren.

```
def dot(dictA, dictB):
    """
    >>> dot({'a':1, 'b':2, 'c': 3}, {'a':4, 'c': 6})
    22
    """




    return
```

(b) Vervollständigen Sie den Programmcode zur Berechnung der Kosinus-Ähnlichkeit zweier Dokumentvektoren (verwenden Sie die Funktion `dot` aus der vorhergehenden Aufgabe; wenden Sie **keine** TF-IDF Gewichtung an).

```
def cosine(dictA, dictB):
    """
    >>> cosine({'a':1, 'b':2, 'c': 3}, {'a':4, 'c': 6})
    0.8153742483272114
    """




    return
```

6 PUNKTE

Aufgabe 4 Git

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

- Erklären Sie kurz (jeweils ein Satz), die Funktion der folgenden git-Befehle:
  (a) `git pull`
  (b) `git add .`
  (c) `git push`
  (d) `git commit -m "Solution to exercise 2."`
- In welcher Reihenfolge wenden Sie die oben angebenen Befehle sinnvollerweise an, wenn Sie eigene Änderungen zum Gitlab remote hinzufügen wollen, und eines Ihrer Team-Mitglieder möglicherweise auch Änderungen vorgenommen hat? Begründen Sie die gewählte Reihenfolge.

4 PUNKTE

NAME:

Aufgabe 5 Naive Bayes Klassifikator

(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 Parameter `smoothing` (Float).

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




  return
```

(b) Vervollständigen Sie die Funktion `sentence_log_probability`, die die logarithmierte Wahrscheinlichkeit $\log P(sentence)$ eines Satzes `sentence` (Liste von Strings) berechnet. Das Dictionary `word_to_count` enthält alle Wörter des Vokabulars als Keys, und die Anzahl der entsprechenden Wortvorkommen im Korpus als Values.
Wie oben gibt `smoothing` (Float) den Glättungsparameter an.
Sie können `log_probability` von Teil (a) der Aufgabe verwenden (berechnen Sie `vocab_size` und `total` aus dem Dictionary `word_to_count`).

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




  return
```

5 PUNKTE

NAME:

Aufgabe 6 K-nearest neighbors

Gegeben eine Liste mit den Dokumentkategorien und deren Kosinus-Ähnlichkeiten. Implementieren Sie folgende Funktionen:

1. `order_nearest_to_farthest` soll die Liste nach der Kosinus-Ähnlichkeit sortieren (vom ähnlichsten Dokument bis zum wenig ähnlichsten).
2. `labels_k_closest` soll die Liste der `k` ähnlichsten Kategorien zurückgeben.

```
def order_nearest_to_farthest(distances):
    """
    >>> order_nearest_to_farthest([(0.2,"news"),(0.5,"cars"),(0.7,"sport")])
    [(0.7,"sport"),(0.5,"cars"),(0.2,"news")]
    """
```

```
def labels_k_closest(sorted_distances,k):
    """
    >>> labels_k_closest ([(0.7,"sport"),(0.5,"cars"),(0.2,"news")], 2)
    [sport, cars]
    """
```

4 PUNKTE

NAME:

Aufgabe 7 K-Means

Gegeben eine Liste mit den Dokumentvektoren (`inputs`) und deren Clusters (`assignments`). Berechen Sie den Cluster Zentroid (`vector mean`) für den Cluster `i`.

```
def vector_mean(inputs, assignments, i):
    import numpy as np
    """
    >>> vector_mean([[1,2,3],[4,5,6],[7,8,9]], [0,0,1], 0)
    [2.5, 3.5, 4.5]
    """
```

4 PUNKTE