



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. Conditional Frequency Distribution	6	
6. Statistics	4	
7. Web Crawling	4	
Summe	30	
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.

```
# Beispiellargumente 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

NAME:

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 angegebenen Befehle sinnvollerweise an, wenn Sie eigene Änderungen zum Gitlab remote hinzufügen wollen, und eines Ihrer Teammitglieder möglicherweise auch Änderungen vorgenommen hat? Begründen Sie die gewählte Reihenfolge.

4 PUNKTE

NAME:

Aufgabe 5 Conditional Frequency Distribution

Gegeben folgender Programmcode:

```
import nltk
from nltk.corpus import udhr

def fct1(list_param, dict_param):
    return nltk.ConditionalFreqDist((language, char_bigram)
                                     for language in list_param
                                     for word in dict_param[language]
                                     for char_bigram in nltk.bigrams(word.lower()))

def fct2(cfd_param, string_param):
    max_score = 0
    for condition in cfd_param.conditions():
        counter = 0
        for word in string_param.split():
            word = word.lower()
            for char_bigram in nltk.bigrams(word):
                counter = counter + cfd_param[condition].freq(char_bigram)
        if counter > max_score:
            max_language = condition
            max_score = counter
    return max_language

languages = ['English', 'German_Deutsch']
language_base = dict((list_item, udhr.words(list_item + '-Latin1'))
                     for list_item in languages)
language_model_cfd = fct1(languages, language_base)
text = "Peter had been to the office before they arrived."
print(fct2(language_model_cfd, text))
```

Erklären Sie kurz den Zweck jeder Methode und des Hauptprogramms.

(a) main:

(b) fct1:

(c) fct2:

6 PUNKTE

NAME:

Aufgabe 6 Statistics

Gegeben ein tokenizierter Text, der durch Liste von Tokens repräsentiert ist. Implementieren Sie folgende Funktionen:

1. `num_tokens` soll die Anzahl von Tokens berechnen
2. `vocabulary_size` soll die Vokabulargröße berechnen
3. `av_wordlength` soll die durchschnittliche Wortlänge berechnen

```
import nltk
from nltk.corpus import gutenber
```

```
def num_tokens(text):
    #returns number of tokens
```

```
def vocabulary_size(text):
    #returns vocabulary size
```

```
def av_wordlength(text):
    #returns average word length
```

```
text = nltk.Text(gutenberg.words("melville-moby_dick.txt"))
print(num_tokens(text))
print(vocabulary_size(text))
```

4 PUNKTE

NAME:

Aufgabe 7 Web Crawling

Implementieren Sie die Funktion `get_raw`, die den bereinigten Text aus der gegebenen URL extrahiert, indem alle HTML markups entfernt werden.

```
import urllib
import bs4
```

```
def get_raw(url):
    #returns the stripped raw text (no HTML markup) from the given URL
```

```
raw = get_raw("http://www.bbc.com/news/world-middle-east-42412729")
```

4 PUNKTE