מבוא לתכנות 52304;

תרגיל 5 - קריאה מקבצים, מילון ואלגוריתמים

להגשה בתאריך XX בשעה YY

שימו לב: בתרגיל זה עבודה בזוגות.
הוראות מפורטות לגבי הגשה בזוגות תוכלו למצוא בתחתית קובץ זה.

המשימה:
בתרגיל זה נפתח כלי לזיהוי שפה מתוך טקסט המבוסס על אותיות לטיניות (למשל, אנגלית מול צרפתית). דוגמא לשימוש בכלי כזה תוכלו למצוא ב Google Translate המזהה באופן אוטומטי את השפה של הביטוי המוקלט. דוגמא נוספת היא זיהוי "שפה" במובן הרחב יותר כך שנוכל להבחין בין כתב שיקספירי לכתב של מחזאי אחר. כלים מעין אלו (משוכללים יותר ממה שנכתוב בתרגיל) שימשו לגילוי טקסטים גנובים וכן לניתוח סגנונות שונים בתנ"ך.

תאור כללי של האלגוריתם:
הרעיון הבסיסי הוא לשמור סטטיסטיקה של אותיות או רצפים של אותיות ולזהות את השפה על סמך ההבדל בין סטטיסטיקות אלו. לדוגמא, האות 'a' מופיעה גם באנגלית וגם בצרפתית במעט מעל ל-8% מהמקרים ולא עוזרת להבחנה בין השפות. לעומת זאת, האות 'f' מופיעה באנגלית במעל ל-2% מהמקרים ואילו בפחות מ-1% מהמקרים בצרפתית. כלומר, אם נראה טקסט באורך מיליון אותיות, נצפה בממוצע ללפחות מ-10,000 'f' בצרפתית ולמעלה מ-20,000 באנגלית. לפיכך, אם למשל בטקסט ספציפי באורך מיליון תווים נצפה ב-18,374 פעמים באות 'f', ננחש שהטקסט לקוח מהשפה האנגלית.

ברצפי אותיות באורך n יש בדרך כלל מידע רב יותר מאותיות בודדות (לרצפים אלו קוראים n-grams). למשל, הרצף באורך 2 הנפוץ ביותר באנגלית הוא 'th' ומופיע ביותר מ-1.5% מהמקרים. בצרפתית, רצף זה הוא אפילו לא אחד מ-30 הנפוצים ביותר! אולם, שימו לב שחישוב על סמך רצפים באורך 2 הוא מסובך יותר (יש יותר רצפים אפשריים כאלו) מאשר חישוב שמבוסס על סמך אותיות בלבד.

בתרגיל זה נאמץ גישה משולבת בו אנו מנסים לזהות שפה קודם כל על סמך אותיות בודדות. אם אנו לא מספיק בטוחים בבחירה, נעבור לרצפים באורך 2, וכן הלאה, עד שנהיה מספיק בטוחים או שנגיע לאורך מקסימלי מותר (ראו פרטים בהמשך). כלומר, האלגוריתם יראה כך:

שלב הלימוד/אימון: נקרא טקסט בכל אחת משתי השפות, נבנה את מילון השכיחויות של כל ה n-grams עד למספר מסוים ונשמור את התוצאה בקובץ.

שלב התחזית:

1) נקרא טקסט בשפה לא ידועה (אחת משתי השפות עליהן התאמנו)
2) עבור n=1,2,3,...K
   a) נחשב את השכיחות של ה n-gram בטקסט
   b) נחשב את המרחק בין השכיחויות של הטקסט לאלו של שתי השפות
   c) אם קיבלנו מרחק מספיק קטן מהשפות לעומת השנייה, נחזיר ניחוש
   d) אחרת, נמשיך בלולאה

להלן מתוארות הפונקציות אותן עליכם לממש. מומלץ לקרוא את התרגיל עד סופו על מנת להבין את מרכיביו ורק אז להתחיל לממש את הפונקציות השונות. אלא אם כן מצוין אחרת, ניתן להניח כי הקלט לפונקציות תקין.

חלק א: n-grams

את הפונקציות הבאות עליכם לכתוב בתוך קובץ ngrams.py. מותר לממש פונקציות עזר נוספות בקובץ אם רוצים אבל חובה לוודא שהפונקציות המפורטות למטה מוגדרות בדיוק כפי שהן מוגדרות בקובץ שלכם.

1) חישוב השכיחויות. ממשו את הפונקציה

compute_ngram_frequency(text, n)

המקבלת טקסט כמחרוזת ארוכה אחת, את המספר n, ומחזירה מילון בו המפתחות הם מחרוזות באורך n והערכים את השכיחות היחסית של המחרוזות. שימו לב שיש לדלג על רווחים, פסיקים או נקודות. אותיות גדולות צריכות להיות מומרות לאותיות קטנות. ניתן להשתמש ב string.lower(), ולהניח שאחרי כל פסיק או נקודה יש רווח או שורה חדשה.

לדוגמא: אם הטקסט הוא "banana pack" ו-n הוא 1 אז המילון שהפונקציה תחזיר יהיה:

{ “b”: 0.1, “a”: 0.4, “n”: 0.2, “p”: 0.1, “c”: 0.1, “k”: 0.1 }

אם הטקסט הוא :

"Hello, banana pack."

אז הפונקציה תחזיר (עבור הערך 1):

{'c': 0.066, 'b': 0.066, 'n': 0.133, 'o': 0.066, 'a': 0.266, 'l': 0.133, 'h': 0.066, 'k': 0.066, 'p': 0.066, 'e': 0.066}

(לשם קיצור, המספרים המוצגים למעלה הם עד דיוק נמוך של ספרות אחרי הנקודה.)

שימו לב שסדר האיברים במילון אינו חשוב. כמו כן מילון תקין הוא כזה שסכום כל הערכים בו שווה ל-1.

2) חישוב שכיחויות עבור **n=1...K.** ממשו את הפונקציה

compute_ngrams_frequency(text, k)

המקבלת טקסט כמחרוזת ארוכה אחת, את המספר k, ומחזירה רשימה של מילונים כך שהאיבר ה-i ברשימה מכיל את מילון ה-ngram עבור n=i+1. לדוגמא, עבור הדוגמא הראשונה לעיל ו-k=2 נקבל רשימה בה האיבר הראשון שווה למילון בדוגמא לעיל, והאיבר השני שווה ל:

{'ck': 0.125, 'na': 0.25, 'an': 0.25, 'ba': 0.125, 'pa': 0.125, 'ac': 0.125 }

שימו לב שבחישוב צריכים להיכלל רק רצפי אותיות בתוך מילה. רצף שבו האות הראשונה נמצאת בסוף מילה אחת והאות השנייה נמצאת במילה שלאחריה לא תיכלל בחישוב.

3) תרגום מילון למחרוזת. כפי שראיתם בתרגול, יהיה נוח להמיר אינפורמציה לטקסט ורק אז לכתוב אותה לקובץ. לצורך זאת, עליכם לכתוב פונקציה

ngram_dict_to_string

המקבלת מילון n-gram כמתואר בסעיף 1, ויוצרת ייצוג של מחרוזת עבורו. במחרוזת מופיע מפתח במילון, מיד לאחריו התו ":" ומיד לאחריו הערך התואם. לאחר מכן מופיע רווח, וזוג המפתח וערך הבאים במילון וכן הלאה. לדוגמא, המחרוזת התואמת למילון

{ “b”: 0.1, “a”: 0.4, “n”: 0.2, “p”: 0.1, “c”: 0.1, “k”: 0.1 }

היא

"b:0.1 a:0.5 n:0.2 p:0.1 c:0.1 k:0.1”

4) עליכם לממש את הפונקציה string_to_ngram_dict המקבלת מחרוזת ומחזירה מילון, כלומר מבצעת את הפעולה ההפוכה בדיוק לפונקציה הקודמת.

5) שמירת רשימת מילונים בקובץ. כתבו פונקציה

write_list_of_ngram_dicts(list_of_dicts, filename)

אשר מקבלת רשימת מילונים (ראו פונקציה compute_ngrams_frequency) וכותבת אותם לקובץ בשם filename. בעת כתיבה לקובץ, נרשום מילון אחד בכל שורה כך שמספר השורות בקובץ שווה לגודל הרשימה של המילונים.

6) טעינת רשימת מילונים מקובץ. כתבו פונקציה

load_list_of_ngram_dicts(filename)

המקבלת שם קובץ ומחזירה רשימת מילונים המופיעים ב filename (כפי שנשמרו על ידי הפונקציה הקודמת).

חלק ב': סיווג שפה

בחלק זה נשתמש בפונקציות שבנינו בחלק הקודם על מנת ללמוד מסווג המפריד בין שתי שפות. עליכם לממש את הפונקציות הבאות בקובץ language_classifier.py. מותר לממש פונקציות עזר נוספות בקובץ אבל חובה לוודא שהפונקציות המפורטות למטה מוגדרות בדיוק כפי שהן מוגדרות בקובץ שלכם.

7) בניית מילון לשתי שפות. עליכם לממש את הפונקציה

build_language_model(language1_filename, language2_filename, k)

המקבלת שני קבצי טקסט ומחזירה רשימה, המכילה שתי רשימות של מילונים עבור n-grams עד גודל k (ראו פונקציה מספר 2 לעיל). שימו לב לדגשים הבאים:

- כל קובץ מכיל מספר שורות טקסט. עליכם להשתמש בטקסט כולו על מנת לחשב את המילון. לדוגמא עבור הקובץ המצורף wiki_text.txt הרשימה שתווצר הוא (עבור k=1):

h': 0.037300177619893425, 'r': 0.06394316163410302, 't': 0.10301953818827708, 'g': '}]
0.012433392539964476, 'u': 0.021314387211367674, 'o': 0.07815275310834814, 'b':
0.010657193605683837, 'p': 0.015985790408525755, 'w': 0.008880994671403197, 'v':
0.007104795737122558, 'a': 0.08703374777975133, 'y': 0.021314387211367674, 'x':
0.007104795737122558, 'd': 0.03374777975133215, 'l': 0.03019538188277087, 'i':
0.08348134991119005, 'z': 0.0017761989342806395, 'e': 0.12788632326820604, 'm':
0.028419182948490232, 'n': 0.07815275310834814, 's': 0.0674955595026643, 'f':
[{0.037300177619893425, 'c': 0.037300177619893425

- אם אחד או יותר מהקבצים לא קיים הפונקציה תחזיר None. ניתן להשתמש בפקודה os.path.isfile() מתוך המודול os.
- אם אחד או יותר מהקבצים אינו מכיל טקסט הפונקציה תחזיר None

8) מרחק בין שני מילוני **n-grams**. ממשו את הפונקציה

compute_ngram_distance(dict1, dict2)

המקבלת שני מילוני עבור n-gram מסוים (ראו פונקציה compute_ngram_freqency) ומחזירה מספר שהוא סכום ההפרשים (בערך מוחלט) בין השכיחויות השונות בין המילונים. אם במילון מסוים מופיע רצף ובמילון השני לא, ההפרש שווה לערך במילון בו הוא מופיע. לדוגמא: המרחק בין שני המילונים הבאים הוא 0.2 + 0.1 + 0.1 = 0.4 עבור ההפרשים בין השכיחויות של האות "n" ,"b", ו "p" בהתאמה (השכיחויות עבור שאר האותיות זהות)

{ “b”: 0.1, “a”: 0.4, “n”: 0.2, “p”: 0.1, “c”: 0.1, “k”: 0.1 }

{ “b”: 0.3, “a”: 0.4, “n”: 0.1, “c”: 0.1, “k”: 0.1 }

9) חיזוי שפה של טקסט. ממשו את הפונקציה

classify_language(text_to_classify, list_of_dicts1, list_of_dicts2, threshold)

המקבלת קובץ טקסט ושתי רשימות מילונים (שנוצרו על ידי הפונקציה (7) ומסווגת את הטקסט על פי השיטה הבאה:

1) טוענים את הטקסט לסיווג ומאתחלים n=1
2) בונים מילון n-gram עבור הטקסט וערך ה-n הנוכחי. מומלץ לשם כך לכתוב פונקציית עזר.
3) מחשבים את המרחק היחסי בין השפות על פי הנוסחה relative_distance = dist1/(dist1 + dist2) כאשר dist1 הוא המרחק בין המילון של הטקסט לבין השפה הראשונה ו- dist2 הוא המרחק בין המילון של הטקסט לבין השפה השנייה. שימו לב שאם dist1=0 אז הטקסט תואם בדיוק לשפה הראשונה והמרחק היחסי יהיה שווה ל-0. לעומת זאת אם dist2=0 אז הטקסט יהיה תואם לשפה השנייה והמרחק יהיה שווה ל-1.
4) אם המרחק היחסי קטן מ-threshold אז הפונקציה תחזיר 1 (המייצג חיזוי של השפה הראשונה), אחרת אם המרחק היחסי המשלים (1-relative distance) קטן מה threshold, תחזיר 2 (המייצג חיזוי של השפה השנייה). שימו לב ש threshold חייב להיות קטן מ 0.5 (ניתן להניח זאת בכתיבת הקוד).
5) אחרת, אם אף תנאי ב 4 לא מתקיים, ואם ברשימת המילונים של השפה קיים מילון עבור n גדול מ-n הנוכחי, קדם n באחד וחזור ל-(2). אם לא קיים מילון כזה, הפונקציה תחזיר 0 (אין חיזוי).

אפשר להניח ששתי הרשימות list_of_dicts1, list_of_dicts2 הן באותו אורך.

חלק ג׳: הפעלה

בחלק זה נשתמש בפונקציות שבנינו בחלקים הקודמים על מנת לבצע את האימון והסיווג בפול. עליכם לממש את הפונקציות הבאות בקובץ run_language_classifier.py. מותר לממש פונקציות עזר נוספות בקובץ אבל חובה לוודא שהפונקציות המפורטות למטה מוגדרות בדיוק כפי שהן מוגדרות בקובץ שלכם.

10) אימון מודל של שפה. כתבו פונקציה בשם train_language_model אשר לא מקבלת ערכים ומבצעת את הפעולות הבאות:

א) הפונקציה תדפיס את ההודעה (עם רווח בודד לאחר הנקודותיים)

Please enter the filename for language 1:

ותחכה שהמשתמש יזין שם קובץ.

ב) הפונקציה תדפיס את ההודעה (עם רווח בודד לאחר הנקודותיים)

Please enter the filename for language 2:

ותחכה שהמשתמש יזין שם קובץ.

ג) הפונקציה תדפיס את ההודעה הבאה (עם רווח בודד לאחר הנקודותיים) :

Please enter the maximal length of the n-gram:

ותחכה שהמשתמש יזין מספר שלם.

ד) הפונקציה תחשב רשימת מילונים עבור שתי השפות ותשמור אותם בשני קבצים ששמם כמו שם קבצי הטקסטים של השפה בתוספת סיומת "dict."

לדוגמא, אם תפעילו את הפונקציה אל שני הטקסטים שבקבצים dutch_training.txt ו- english_training.txtתקבלו קבצי מילון
שזהים בתוכנים לקבצים שצורפו לתרגיל dutch_training.dict ו- english_training.dict.

11) סיווג טקסט לא ידוע. כתבו פונקציה בשם classify_unknow_text אשר לא מקבלת ערכים ומבצעת את הפעולות הבאות:

א) הפונקציה תדפיס את ההודעה (עם רווח בודד לאחר הנקודותיים)

Please enter the filename for language 1:

ותחכה שהמשתמש יזין שם קובץ (קובץ טקסט ולא מילון).

ב) הפונקציה תדפיס את ההודעה (עם רווח בודד לאחר הנקודותיים)

Please enter the filename for language 2:

ותחכה שהמשתמש יזין שם קובץ (קובץ טקסט ולא מילון).

ג) הפונקציה תדפיס את ההודעה (עם רווח בודד לאחר הנקודותים)

Please enter the filename to classify:

ותחכה שהמשתמש יזין שם קובץ. אם תוזן המילה exit (בדיוק, באותיות קטנות),
הפונקציה תעצור, ותחזיר None. אם אחד משלושת הקבצים שהמשתמש הקליד אינו קיים, הפונקציה תדפיס (ללא רווח בסוף):

File error: some files do not exist.

ותחזור ל א).

ד) הפונקציה תדפיס את ההודעה הבאה (עם רווח בודד לאחר הנקודותיים) :

Please enter the maximal length of the n-gram:

ותחכה שהמשתמש יזין מספר שלם.

ה) הפונקציה תדפיס (עם רוח בודד לאחר הנקודותיים):

Please enter threshold value between 0 and 0.5:

ותחכה שהמשתמש יזין מספר (אפשר להניח שהקליד מספר בין 0 ל 0.5).

ו) הפונקציה תנסה לסווג את הטקסט (ראו פונקציית classify_language) ותדפיס את אחת ההודעות הבאות לפי התוצאה:

I believe the text matches the language in language_filename_X

כאשר language_filename_X הוא שם הקובץ של השפה המתאימה. אם פונקציית הסיווג לא החזירה סיווג, תודפס ההודעה (ללא רווח בסוף)

Sorry, I could not figure this one out!

ז) הפונקציה תחזור לשלב (א)

לדוגמא, אם ננסה לסווג את הטקסט בקובץ wiki_text.txt, על סמך הטקסטים English_training.txt, dutch_training.txt, עם 3 כמספר מקסימלי של ngrams, וערך threshold של 0.4, נקבל בסוף את הפלט הבא:

I believe the text matches the language in english_training.txt

. אנו ממליצים להתחיל לעבוד עליו מוקדם שכן התרגיל ארוך.

הנחיות כלליות להגשה

- הנכם רשאים להגיש תרגילים דרך מערכת ההגשות באתר הקורס מספר רב של פעמים. ההגשה האחרונה בלבד היא זו שקובעת ושתיבדק.

- לאחר הגשת התרגיל, ניתן ומומלץ להוריד את התרגיל המוגש ולוודא כי הקבצים המוגשים הם אלו שהתכוונתם להגיש וכי הקוד עובד על פי ציפיותיכם.
- ניתן לאחר בהגשת התרגיל עד שלושה ימים, אך זה יהיה כרוך בניכוי ציון, כפי שמפורט באתר הקורס ובמסמך נהלי הקורס.
- קראו היטב את קובץ נהלי הקורס לגבי הנחיות נוספות להגשת התרגילים.
- שימו לב - יש להגיש את התרגילים בזמן!

בהצלחה!