עיבוד שפות טבעיות סמסטר א' תשפ"א
תרגיל בית מספר 4: **וקטורי מילים**
מועד הגשת התרגיל: 2.1.2021 בשעה 23:59

בתרגיל זה נעסוק בייצוג טקסט באמצעות וקטורי מילים (word embeddings). המשימה העיקרית בתרגיל זה היא בעלת אופי דומה למשימה שניתנה בתרגיל הקודם אך יהיו שינויים בהגדרות אשר יפורטו בהמשך.

בתרגיל זה ניצור שני מודלים של וקטורי מילים, בהם המילים ייוצגו ע"י וקטורים בשני אורכים שונים (50, 300). **המודלים** יתבססו על קובץ של word embeddings של GloVe: Global Vectors for Word Representation, ובו ייצוג של 400,000 מילים באנגלית. הוקטורים אומנו על טקסטים מויקיפדיה ו Gigaword. הקובץ זמין להורדה כאן – יש להוריד קובץ בשם: glove.6B.zip. קובץ ה-zip שתורידו כולל 4 קבצים שמכילים וקטורי מילים באורכים שונים – כל מילה מיוצגת ע"י וקטור. אנחנו נשתמש בקבצים שמכילים וקטורים באורך 50 ובאורך 300. שמות הקבצים הם glove.6B.50d.txt ו- glove.6B.300d.txt, בהתאמה. את קובץ הטקסט נמיר לקובץ בפורמט word2vec תוך שימוש בספריה **gensim** ובמודול **glove2word2vec**. פקודת ההמרה היא:

```
from gensim.scripts.glove2word2vec import glove2word2vec
glove2word2vec(glove_file, word_to_vec_file)
```

כאשר glove_file הוא אחד מהקבצים שהורדתם (עם וקטורים בממד 50 או 300) ו-word_to_vec_file הוא קובץ הפלט שאתם תקבעו את מיקומו ואת שמו.

יש לייצר שני קבצי word_to_vec עבור שני האורכים. שני הקבצים הללו יהוו קלט לתכנית שתכתבו (אין צורך להגיש אותם, לצורך הבדיקה אשתמש בקבצים מוכנים אשר מפאת גודלם לא אוכל להעלותם לאתר הקורס). הקוד שממיר את קובץ ה glove לפורמט word2vec איננו להגשה.

כדי לטעון את הקבצים השתמשו בספריה ***gensim*** ובמודול ***KeyedVectors*** . הפקודה לטעינת הקובץ היא:

```
from gensim.models import KeyedVectors
word2vec_pre_trained_model_50 = KeyedVectors.load_word2vec_format(MODEL_FILE_50,
binary=False)

word2vec_pre_trained_model_300 = KeyedVectors.load_word2vec_format(MODEL_FILE_3-0,
binary=False)
```

כאשר **MODEL_FILE_50** ו- **MODEL_FILE_300** הם שני ה paths לקבצי ה word embeddings (עם וקטורים במימד 50 ו 300 בהתאמה) שיצרתם.

שלב א' – הכנה והתנסות

לפני התחלת התרגיל, נרצה להתנסות מעט ולקבל תחושה לגבי המודלים שברשותנו ומה ניתן להפיק מהם. השתמשו בפונקציה similarity שמקבלת שתי מילים ומחשבת את הדמיון ביניהן – כלומר, את קוסינוס הזווית בין הוקטורים שמייצגים אותן. הסתכלו בתוצאות שהפונקציה מחזירה עבור זוגות מילים שאנחנו חושבים עליהן כקרובות לעומת מילים שאין ביניהן קשר לדעתנו (למשל בין צמד המילים apple, beautiful לעומת צמד המילים beautiful, ugly). בדקו אם התוצאות דומות למה ששיערתם. השוו בין הערכים המתקבלים בכל אחד מהמודלים שברשותכם. בחרו 4 צמדי מילים שהקשר ביניהן נראה לכם מעניין, הוסיפו את תוצאת חישוב המרחק לדוח, ודווחו האם התוצאות תואמות את הציפיות שלכם.

ניתן לקבל את רשימת המילים הדומות ביותר למילה נתונה ע"י קריאה לפונקציה most_similar. בחרו 4 מילים וכתבו בדוח את סט התוצאות שקיבלתם לכל מילה. נסו לשער ממה נובעים ההבדלים בתוצאות, אם יש כאלה.

בנוסף, אפשר לבצע פעולות על הוקטורים ולקבל תוצאות מעניינות, למשל:

girl=woman+(boy-man)

הקוד לביצוע החישוב נראה כך:

```
word2vec_model.most_similar(positive=['boy','woman'], negative=['man'])
```

הראו דוגמה לשלושה "תרגילים" שכאלה, בהם מתקבלת מילה שהיא לדעתכם תוצאה סבירה. השוו בין התוצאה שהתקבלה בפועל לבין התוצאה שציפיתם לקבל ונסו לחשוב מה מקור ההבדלים אם יש כאלה.

שלב ב' - סיווג

בתרגיל זה נחזור על משימת סיווג השפה:

- בתרגיל הקודם איזנתם את כמות הטקסט, כך שלכל מחלקה היה מספר זהה של יחידות לסיווג. בתרגיל הנוכחי תשתמשו באותם קבצי קלט כמו בתרגיל הקודם, אבל כפי שהם וללא איזון – כלומר תהיה כמות טקסט שונה לכל אחת ממחלקות הסיווג.
- כדי לפתור את בעיית חוסר האיזון, השתמשו בטכניקה שנקראת **SMOTE**: **Synthetic Minority Oversampling Technique** . טכניקה זו מייצרת דוגמאות – במקרה שלנו משפטים – סינטטיות ע"מ להגדיל את קבוצת המיעוט (במקרה שלנו – אנגלית פשוטה) ולהשוות את מספר הדוגמאות בה למספר בקבוצת הרוב (במקרה שלנו אנגלית סטנדרטית). הטכניקה הזו ממומשת בספריית imbalanced-learn (תיעוד זמין כאן).
- שינוי נוסף הוא גודל יחידת הסיווג. במקום משפט בודד, גודל יחידת הסיווג יהיה כעת זוגות של משפטים (chunk בגודל 2).
- למרות השימוש ב SMOTE, למסווג עדיין יש יותר מידע על קבוצת הרוב (אנגלית סטנדרטית), ולכן **לא מדויק** יהיה לומר שניחוש אקראי יפיק דיוק של 50%. לכן, לשם הערכה יותר טובה של ביצועי המסווגים תידרשו להשתמש במדדים נוספים/אחרים להערכת איכות התוצאות. המדדים הדרושים מפורטים בהמשך.

השתמשו במסווג random forest, שממומש בספריה sklearn. בנו את ה-feature vectors על בסיס ה- word embeddings כך שה-feature vector של כל צמד משפטים הוא הממוצע המשוקלל של ערכי הוקטורים של כל המילים ב- chunk, לפי הנוסחה הבאה:

$$\left(\sum_i w_i \times v_i\right)/k$$

w_i - הוא המשקל של המילה ה-i (סקלאר, שייקבע באופן שמוסבר בהמשך)

v_i - הוא הוקטור שמייצג את המילה ה- i במודל ה-word embeddings

k – מספר המילים ב-chunk *(*צמד המשפטים שמהווים את יחידת הסיווג)

אם המשפט כולל מילה שלא נמצאת במודל שטענתם, התעלמו ממנה ואל תשקללו אותה בחישוב. עבור כל אחד מהמודלים –המודל שבו וקטורים באורך 50, והנוסף שבו וקטורים באורך 300 – אמנו את המסווג ב 3 דרכים שונות:

1. המשקל לכל מילה הוא 1, כלומר $\forall i\ w_i = 1$ (ממוצע חשבוני)
2. הגרילו משקלים אקראיים בטווח (0,5) לכל מילה במשפט.
3. חשבו על שיטה משלכם לקביעת המשקלים שתביא לתוצאות סיווג טובות יותר (למשל – מילה ראשונה או אחרונה במשפט צריכה לקבל משקל גבוה יותר, מילים נפוצות מאוד יקבלו משקל נמוך יותר וכו'). שימו לב שאין חסמים על המשקולות. פרטו את השיטה שהשתמשתם בה בדוח שתגישו.

לצורך הערכת הביצועים נרצה גם במשימה זו להשתמש בשיטה של 10-fold cross validation. לצורך כך השתמשו ב-sklearn.model_selection.cross_val_score בעזרת sklearn.model_selection.StratifiedKFold. לכל אחד מהמודלים ולכל אחת משיטות המשקול המפורטות מעלה, דווחו על מדדי הערכת הביצועים הבאים (שהוסברו בהרצאות):

1. Accuracy
2. Precision
3. Recall
4. f1

אפשר להיעזר במודול sklearn.metrics לצורך מימוש המדדים.

הוראות הגשה:

- הגישו את הקבצים הבאים:
  - קובץ קוד בשם hw4.py
  - אם דרושים לכם קבצים נוספים על מנת שהקוד יהיה מסודר, אתם יכולים להוסיף אותם – רק שימו לב שהריצה תקינה ושלא חסר דבר.
  - דוח בפורמט PDF בשם hw4Report.pdf.

- על הקוד שמבצע את הסיווג להיות מופעל משורת הפקודה בעזרת הפקודה:

python hw3.py <input_dir> <word2vec_50> <word2vec_300> <output_path>

כאשר:

  - input_dir היא התיקייה שבה נמצאים הקבצים למשימת הסיווג (אותם הקבצים שהשתמשתם בהם למשימת סיווג לפי שפת האם בתרגיל הקודם)
  - word2vec_50 הוא הנתיב והשם של קובץ ה-word embeddings שאומן מראש ובו וקטורי מילים במימד 50.
  - word2vec_300 הוא הנתיב והשם של קובץ ה-word embeddings שאומן מראש ובו וקטורי מילים במימד 300.
  - output_path הוא השם והנתיב של קובץ פלט שיכיל את תוצאות הסיווגים, על פי הפורמט המוגדר בהמשך.
- על התכנית לכתוב לקובץ הפלט את תוצאות הסיווג , עבור שני מודלי ה-word2vec, עבור כל אחת משיטות המשקול. יש להדפיס את כל המדדים שחושבו בדיוק של 2 ספרות אחרי הנקודה. הפלט צריך להיראות כך:

Arithmetic mean:
word2vec_50 model performance:
Accuracy : <>
Precision: <>
Recall: <>
F1:<>
word2vec_300 model performance:
Accuracy: <>
Precision: <>
Recall: <>
F1:<>

Random weights:
word2vec_50 model performance:
Accuracy: <>
Precision: <>
Recall: <>
F1:<>
word2vec_300 model performance:

Accuracy : <>
Precision: <>
Recall: <>
F1:<>

My weights:
word2vec_50 model performance:
Accuracy: <>
Precision: <>
Recall: <>
F1:<>
word2vec_300 model performance:
Accuracy: <>
Precision: <>
Recall: <>
F1:<>

הדו"ח שאתם מגישים צריך לכלול:

- פירוט של תוצאות ההתנסות שמפורטות בחלק א'.
- הסבר של שיטת המשקול שחשבתם עליה לצורך אימון המסווגים.
- השוואה בין תוצאות הסיווג של שיטות המשקול השונות ושני המודלים השונים. האם אתם רואים הבדלים גדולים בתוצאות בשיטות משקול שונות? האם ישנם הבדלים משמעותיים בתוצאות הסיווג בין שני המודלים?

יש להקפיד על עבודה עצמאית. צוות הקורס יתייחס בחומרה להעתקות או שיתופי קוד.
תאריך הגשה: 2.1.21 , עד השעה 23:59.
שאלות על התרגיל אפשר לשאול בפורום תרגילי בית.

בהצלחה!