תרגיל 4 – Word Embeddings

מבוא

בתרגיל הזה נתנסה בכמה שימושים שיש ל-word embeddings: דמיון מילים, מודלי שפה וסיווג. גם בתרגיל זה נשתמש בקורפוס הכנסת, כלומר בקובץ הcsv שיצרתם בתרגילים הקודמים.

חלק א': יצירה של מודל Word2Vec ואימונו על קורפוס הכנסת

בחלק זה, אתם תבנו מודל word2vec שיתאמן על המשפטים מקורפוס הכנסת, ויוכל ליצור בעזרתם word embeddings לכל מילה בקורפוס. כלומר, וקטורים שמייצגים כל מילה בקורפוס.

לשם כך, תשתמשו בספרייה gensim ותייבאו ממנה את המודל Word2Vec באופן הבא:

```
from gensim.models import Word2Vec
```

א. הכינו למודל את המשפטים (לאחר טוקניזציה) מתוך קורפוס הכנסת. לשם כך, הסירו מהמשפטים טוקנים שאינם מילים (סימני פיסוק, מספרים וכד') והכינו מהם רשימה של רשימות של טוקנים. כלומר, כל משפט יהיה רשימה של הטוקנים הנמצאים בו, והם יוספו לרשימה המכילה את כל הרשימות האלו, שתהיה רשימת כל המשפטים.
דוגמה לאיך הרשימה אמורה להיראות:

```
# Example: tokenized_sentences = [['היום', 'הוא', 'יום', 'נפלא'], ['אני', 'אוהב', 'עברית']]
```

ב. צרו מודל Word2Vec המייצר וקטורי word embeddings בגודל 100 מתוך משפטי הכנסת, באופן הבא:

```
model = Word2Vec(sentences=tokenized_sentences, vector_size=100, window=5, min_count=1)
```

כדאי גם לשמור את המודל כדי שלא תצטרכו לאמן אותו מחדש בכל ריצה:

```
model.save("knesset_word2vec.model")
```

ג. כעת אתם יכולים להשתמש במודל באופן הבא:

```
word_vectors = model.wv
print(word_vectors['ישראל'])  # Get the vector for the word
```

ענו על השאלות הבאות:

1. היכנסו לתיעוד המודל, הבינו מה המשמעות של כל ארגומנט שנתנו לו (vector_size, window, min_count) והסבירו מה היתרונות והחסרונות של הערכים שנבחרו עבור השימוש שלנו. האם הייתם בוחרים ערכים אחרים?
2. הסבירו מה הבעיות שיכולות לעלות משימוש במודל הנ"ל שאומן על הקורפוס שלנו. התייחסו בתשובתכם לאופן יצירת הקורפוס, לגודל שלו ולשימושים פוטנציאלים של המודל.

חלק ב': דמיון בין מילים

וקטורי word embeddings אמורים לייצג את משמעות המילים. לכן, עבור מילים דומות במשמעות, נצפה שהמרחק בין הוקטורים שלהן יהיה קצר. בחלק זה נבדוק אם זה באמת מתקיים.

א. לפניכם רשימה של מילים, עבור כל מילה ברשימה עליכם למצוא את 5 המילים הכי קרובות אליה (לא כולל עצמה כמובן). אתם יכולים לעשות זאת ע"י שימוש בפונקציה similarity שמחשבת את ה cosine similarity בין שני וקטורי מילים.

```
similarity_score = model.wv.similarity('word1', 'word2')
```

רשימת המילים:

1. ישראל
2. כנסת
3. ממשלה
4. חבר
5. שלום
6. שולחן

הדפיסו את המילים והמרחקים לקובץ בשם "knesset_similar_words.txt" באופן הבא:

<first given word>: (<first word>, <similarity score>), …, (<fifth word>, <similarity score>)

…

<last given word>: (<first word>, <similarity score>), …, (<fifth word>, <similarity score>)

ב. צרו לכל משפט בקורפוס sentence embeddings ע"י ממוצע משוקלל של וקטורי המילים במשפט. התעלמו מטוקנים שWord2Vec לא מכיר בחישוב זה (אלו שהסרנו בחלק א'). החישוב יעשה באופן הבא:

$$\frac{\sum_{i=1}^{k} v_i}{k}$$

כאשר v_i הוא וקטור המילה ה-i במשפט ו-k הוא מספר המילים שהחשבנו במשפט.

ג. בחרו 10 משפטים מהקורפוס, המכילים לפחות 4 טוקנים תקינים (מילים), ועבור כל אחד מצאו בעזרת cosine similarity את המשפט הכי קרוב אליו.

הדפיסו את המשפטים לקובץ בשם "knesset_similar_sentences" באופן הבא:

<first chosen sentence>: most similar sentence: <most similar sentence>

...

<last chosen sentence>: most similar sentence: <most similar sentence>

הערה: הדפיסו את המשפטים המקוריים מהקורפוס שלכם, עם רווח בין הטוקנים (כולל סימני פיסוק וכדו').

ד. לפניכם 4 משפטים שבכל אחד מהם ישנן מילים המסומנות באדום. השתמשו בפונקציה most_similar כדי למצוא מילה קרובה למילה המסומנת באדום והחליפו אותה בה. פונקציה זו מקבלת גם ארגומנטים של positive ו negative שמיועדות לביצוע אריתמטיקה וקטורית, ובעזרתם ניתן לכוון את המודל לבחור במילה יותר מתאימה. בדקו האם אתם מצליחים להחליף את המילים האדומות במילים אחרות, כך שהמשפט נשאר תקין, הגיוני ובעל משמעות דומה. **הסבירו בדו"ח מה ניסיתם, מה עשיתם, מה קיבלתם והאם הצלחתם במשימה.**

דוגמה:

נניח ויש לנו את המשפט הבא:

"היום בבוקר מהנדסת הציגה את המודל שבנתה"

ונניח והשתמשתם בפונקציה most_similar כדי למצוא מילה שתחליף את המילה האדומה "מהנדסת" במילה מספיק קרובה שתשמור על המשמעות וגם על נכונות תחבירית.

תחילה, ניסיתם להשתמש בפונקציה כך:

```
similar_word = model.wv.most_similar('מהנדסת', topn=1)
```

ונניח שקיבלתם חזרה את המילה "מהנדס"

מילה זו לא מסתדרת מספיק טוב במשפט כי הוא מנוסח בלשון נקבה, אז ננסה שוב, הפעם בעזרת הארגומנטים positive, negative:

```
similar_word = model.wv.most_similar(positive=['מהנדסת', 'אישה'],
negative=['גבר'], topn=1)
```

כעת המודל החזיר את המילה "הנדסאית" שמתאימה למשפט בצורה טובה יותר.

באופן דומה, במשפטים הבאים נסו להחליף את המילים האדומות במילה הכי טובה שתצליחו:

1. ברוכים הבאים , הכנסו בבקשה לחדר .
2. אני מוכנה להאריך את ההסכם באותם תנאים.
3. בוקר טוב , אני פותח את הישיבה .
4. שלום , הערב התבשרנו שחברינו היקר לא ימשיך איתנו בשנה הבאה .

הדפיסו את הפלט עם המילים המחליפות שבחרתם, לקובץ בשם "red_words_sentences.txt" בפורמט הבא:

<original sentence>: <new sentence where the red words were replaced with the new words>

...

<original sentence>: <new sentence where the red words were replaced with the new words>

הערות:

1. מותר לכם לקחת אחת מתוך שלושת המילים הכי דומות שהמודל מחזיר topn=3.
2. אף על פי שלרוב זה נחוץ וזאת נקודת התחלה טובה, אינכם חייבים להשתמש במילה האדומה עצמה כדי להגיע למילה מחליפה. למשל, אם המילה האדומה לא קיימת בקורפוס שלכם אבל מילה קרובה כן, או אם אתם חושבים שבעזרת מילה אחרת שקרובה אליה במשמעות תצליחו להביא את המודל להחזיר מילה מחליפה מתאימה יותר, מותר לכם לעשות זאת.

ענו על השאלות הבאות:

1. האם המילים הכי קרובות שקיבלתם בסעיף א' תואמות את הציפיות שלכם? הסבירו. גם אם תאמו לציפיות וגם אם לא, נסו להסביר מדוע זה עבד או לא עבד טוב.
2. אם ניקח שתי מילים שנחשבות להפכים (antonyms), למשל "אהבה" ו"שנאה", או "קל" ו"כבד". האם היינו מצפים שהמרחק בין שני וקטורי המילים שלהן יהיה קצר או ארוך?
3. מצאו זוג מילים שנחשבות להפכים (antonyms) הקיימות בקורפוס שלנו ובידקו את המרחק ביניהן. האם הציפייה שלכם מסעיף 2 מתקיימת עבורן עם המודל שבניתם?
4. האם המשפטים הכי קרובים בסעיף ג' תאמו לציפיות שלכם? הסבירו. גם אם תאמו לציפיות וגם אם לא, נסו להסביר מדוע זה עבד או לא עבד טוב.

חלק ג': סיווג

אמנו מודל KNN בדומה לזה שמימשתם בתרגיל בית 3, על מנת לסווג בין צ'אנקים של משפטי וועדות לבין צ'אנקים של משפטי מליאות. הפעם, במקום וקטורי המאפיינים שהשתמשתם בהם בתרגיל 3, השתמשו בsentence embeddings, כפי שהגדרנו בסעיף בחלק ב' סעיף ב'. כאשר הפעם נחשיב בחישוב הממוצע את כל המשפטים בצ'אנק כאילו היו משפט אחד ארוך. עשו זאת עבור צ'אנק בגודל 1 (כלומר משפט אחד), 3 ו5.

השתמשו בשיטת החלוקה לקבוצת אימון וקבוצת בדיקה כפי שעשיתם בתרגיל 3. הפיקו classification_report וצרפו אותו לדו"ח.

ענו על השאלות הבאות:

1. האם עבור אותם פרמטרים ותנאים שהשתמשתם בהם בתרגיל 3 (צ'אנק בגודל 5, אותה כמות שכנים , שיטת חלוקה וכו') קיבלתם תוצאות טובות יותר או פחות עבור וקטור המאפיינים הנ"ל?
2. עבור התשובה שעניתם בסעיף 1, הסבירו מדוע לדעתכם זה קרה.
3. עבור איזה גודל צ'אנק קיבלתם תוצאות יותר טובות? האם זה נכון גם לגבי וקטורי המאפיינים שהשתמשתם בהם בתרגיל 3? הסבירו.

חלק ד': שימוש במודלי שפה גדולים

בשנים האחרונות, מודלי שפה גדולים (LLM- Large language models) כמו GPT-3/4, BERT, RoBERTa ועוד, תפסו מקום חשוב מאוד בעולם עיבוד השפות הטבעיות. אלו הם מודלים שבנויים מרשתות נוירונים עמוקות מאוד ומאומנים על הרבה מאוד טקסט. הם לומדים "להבין" את המשמעויות העמוקות והמורכבויות שיש בשפה והם מסוגלים לבצע בדיוק גבוה משימות רבות בתחום הNLP כגון חיזוי המילה הבאה במשפט, יצירה (generation) של טקסט קוהרנטי ורלוונטי, תרגום, מענה על שאלות ועוד.

לב ההצלחה של מודלים אלו והבסיס שבעזרתו הם מסוגלים לבצע את כל המשימות האלו, הוא הדרך שבה הם מצליחים לייצר word embeddings. בניגוד לWord2Vec, ה- word embeddings שמודלים אלו מייצרים הם תלויי קונטקסט.
מודלי word embeddings לא קונטקסטואלים יפיקו לאותה מילה תמיד את אותו הוקטור, אף על פי שיכולה להיות לה משמעות שונה לחלוטין בהקשרים שונים. למשל: המילה "חשב" במשפט "הוא חשב שהרעיון היה מצויין" לעומת במשפט "הוא חשב במקצועו".

מודלי שפה גדולים מתייחסים לקונטקסט שסביב המילה כדי לייצר לה וקטור המתאים למשמעות שלה באותו ההקשר.

כדי להתנסות במודל כזה, אנחנו נשתמש במודל שנקרא DictaBERT. זהו מודל מבוסס BERT שאומן על הרבה מאוד טקסטים בעברית ומסוגל לחזות בדיוק יחסית גבוה מילים ממוסכות (masked) במשפט.

מצורף לתרגיל זה קובץ המשפטים הממוסכים שהשתמשתם בו בתרגיל בית 2 masked_sentences.txt.
החליפו את הכוכביות [*] בטוקן [MASK] כפי שdictaBert מצפה, עקבו אחר ההוראות בדף של המודל ותחזו בעזרתו את המילים החסרות במשפטים.

צרו קובץ פלט, בשם dictabert_results.txt בפורמט דומה לזה שעשיתם בתרגיל בית 2, רק עם החיזויים של המודל הזה, באופן הבא:

Original sentence: <The first original sentence as appeared in the sentences.txt file>
DictaBERT sentence: <The sentence with the generated tokens as was produced by the committee LM>
DictaBERT tokens: <A list with the generated tokens, separated by a comma (“,”)>

ענו על השאלות הבאות:

1. האם קיבלתם משפטים הגיוניים? מבחינת התוכן, קוהרנטיות ומבחינה תחבירית.
2. השוו את התוצאות שקיבלתם עכשיו לאלו שקיבלתם בתרגיל בית 2. האם יש שיפור בתוצאות לדעתכם?
3. האם יש משפטים שעבורם המודל עבד פחות טוב? אם כן, הסבירו מה לדעתכם הסיבה לכך. אם לא, האם לדעתכם הוא יעבוד בצורה מושלמת על כל משפט מתוך קורפוס הכנסת? הסבירו.

ספריות מותרות לשימוש

אתם יכולים להשתמש ב

Pandas, Numpy, scikit-learn, genism, transformers*

(* רק עבור חלק ד')

ובכל ספריה סטנדרטית של python.

אתם יכולים לחפש שם של ספריה ב https://docs.python.org/3/library/index.html על מנת לבדוק אם זו ספריה סטנדרטית. לא יהיה מענה על שאלות לגבי שימוש בספריות ספציפיות.

הערות כלליות

1. על הקוד שלכם להיות מסוגל להתמודד עם שגיאות עבור כל שלב בתהליך. השתמשו ב Try-Except blocks לפי הצורך.
2. אתם יכולים לעבוד בכל סביבת עבודה שנוחה לכם, אך הפתרון ייבדק בסביבת windows ועליכם לדאוג שהוא ירוץ בהצלחה בסביבה זו.

אופן ההגשה

1. ההגשה היא בזוגות בלבד.
2. עליכם להגיש קובץ zip בשם hw4_<id1>_<id2>.zip (כאשר <id1>,<id2> הם מספרי תעודות הזהות של הסטודנט הראשון והשני בהתאמה), המכיל את הקבצים הבאים:
   a. קובץ python בשם **knesset_word2vec.py** המכיל את כל הקוד הנדרש כדי לממש את חלקים א'-ב'.

i. - הקלט לקובץ יהיה נתיב לקובץ הcsv של קורפוס הכנסת ונתיב לתיקייה בה תייצרו את שלושת קבצי הפלט (התואמים לקבצים בd,e,f).
- הפלט צריך להיות הקבצים התואמים לd,e,f שמורים בתיקייה שהתקבלה כקלט.

ii. על הקובץ לרוץ תחת הפקודה:

```
python knesset_word2vec.py <path/to/a_corpus_csv_file.csv> <path/to/output_dir>
```

b. קובץ python בשם knesset_word2vec_classification.py המכיל את כל הקוד הנדרש כדי לממש את חלק ג'.

i. - הקלט לקובץ יהיה נתיב לקובץ הcsv של קורפוס הכנסת, נתיב למודל word2vec (כפי ששמרתם בחלק א')
- הפלט יהיה הדפסה למסך של שלושת הclassification reports (עבור כל גודל צ'אנק, כפי שתואר בחלק ג').

ii. על הקובץ לרוץ תחת הפקודה:

```
python knesset_word2vec_classification.py <path/to/a_corpus_csv_file.csv>
<path/to/Knesset_word2vec.model>
```

c. קובץ Python בשם knesset_dictabert.py המכיל את כל הקוד הנדרש כדי לממש את חלק ד'.

i. - הקלט לקובץ יהיה נתיב לקובץ masked_sentences.txt ונתיב לתיקייה בה תייצרו את קובץ הפלט dictabert_results.txt
- הפלט צריך להיות הקובץ dictabert_results.txt שמור בתיקייה שהתקבלה כקלט.

ii. על הקובץ לרוץ תחת הפקודה:

```
python knesset_dictabert.py <path/to/masked_sentences.txt> <path/to/output_dir>
```

d. קובץ text בשם knesset_similar_words.txt כפי שתואר בחלק ב' סעיף א'.

e. קובץ text בשם knesset_similar_sentences.txt כפי שתואר בחלק ב' סעיף ג'.

f. קובץ text בשם red_words_sentences.txt כפי שתואר בחלק ב'. סעיף ד'.

g. קובץ text בשם dictabert_results.txt כפי שתואר בחלק ד'.

h. קובץ הcsv של קורפוס הכנסת איתו עבדתם.

i. קובץ PDF בשם **<id1>_<id2>_hw4_report.pdf** ובו דו"ח המפרט על הקוד, על ההחלטות שקיבלתם במהלך העבודה על התרגיל ומענה על כל השאלות בכל החלקים.
<u>אל תשכחו לציין בתחילת הדו"ח את שמותיכם ותעודות הזהות שלכם.</u>

יש להקפיד על עבודה עצמית, צוות הקורס יתייחס בחומרה להעתקות או שיתופי קוד, כמו גם שימוש בכלי AI דוגמת chatGPT.

ניתן לשאול שאלות על התרגיל בפורום הייעודי לכך במודל.

יש להגיש את התרגיל עד לתאריך 15.3.24 בשעה 23:59.

בהצלחה!