עיבוד שפות טבעיות

תרגיל בית 4

סלאם קייס,327876116

עבאס אסמאעיל,214742025

שלב 1)

סעיף 1)

פו פשוט בניתי שתי פונקציות

הראשונה לוקחת משפט ומסירה מהמשפט הזה כל מילה שהתוו הראשון והתוו האחרון בה ,הוא אינו בעברית(משמעות של זה שהיא מסירה כל מילה שהיא אינה בעברית (מספרים,סימני פסוק)כנדרש)

```
def prepare_one_sent(sent):
    try:
        letter_in_hebrow = [ 'א', 'ב', 'ג', 'ד', 'ה', 'ו', 'ז', 'ח', 'ט', 'י', 'כ', 'ך', 'ל', 'מ', 'נ', 'ו', 'ס', '
        clean_sent=[]
        tokens = sent.strip().split(' ')
        for token in tokens:
            if ( token[-1] in letter_in_hebrow ) and (token[0] in letter_in_hebrow): #if the fist and last letter is
                clean_sent.append(token)
        return clean_sent
    except Exception as e:
        print(f"An error occurred in prepare_one_sent: {e}")
```

הפונקציה השניה היא יותר פשוטה היא לוקחת את כל המשפטים ואז מטפלת בהם בעזרת פונקציה ראשונה ומחזירה את הטקסט הנקי

```
def prepare_sentnces(sentences):
    try:
        cleaned_sentnces=[]
        for sent in sentences:
            cleaned_sentnces.append(prepare_one_sent(sent))

        return cleaned_sentnces

    except Exception as e:
        print(f"An error occurred in prepare_sentnces: {e}")
```

פו פשוט את האימון , אם אני רוצה לאמן אז הופף train_flag ל 1 ורץ אחרת יעשה טעינה למודל קיים.

אם train_flag=1 אז אנחנו מגדירים את המודל ,כמו בשלב 1.2 ,אחרת עושים load

```
train_flag=1
if len(sys.argv) != 3:
    print('Incorrect input, please enter the folder path and the output path.')
    sys.exit(1)
corpus_file_path=sys.argv[1]
output_path=sys.argv[2]
corpus_data=pd.read_json(corpus_file_path, lines=True, encoding='utf-8')
if train_flag == 1:
    prepared_sentences = prepare_sentnces(corpus_data['sentence_text'])
    model =Word2Vec(sentences=prepared_sentences,vector_size=100,window=5,min_count=1)
    model.save(os.path.join(output_path,'knesset_word2vec.model'))
else:
    model_path = "knesset_word2vec.model"
    model= Word2Vec.load(model_path)
```

סעיף 2)

Vector_size : זה פשוט מגדיר את הגודל של הווקטור של כל מילה

בחירת ווקטור גודל בדרך כלל עוזר במציאת יותר מידע על כל מילה אבל לפעמים זה גם גורר ל over fitting באימון(כלומר הוא יהיה מאומן יותר מדי על קורפוס מוסיים בגלל שהוקקטורים ארוכים אבל לא מתמודד טוב עם משהו חדש) וגם מוסיף זמן אימון , ולהפף ווקטור יותר קטו צריך פחות זמן ותופס פחות מאפיינים.

אז בסוף אנחנו עובדים עם קורפוס של קניסת (כמעט מאה אלף שורות)שהוא לא פשוט אבל גם לא מוסבך יותר מדי לכן בחרתי ב vector_size שהוא 100 שהוא משהוא בינוני לא מקבל over_fitting וגם תופס מידע מספיק על המילים

וגם נסיתי לבחור בערכים אחרים אבל למשל כשבחרתי ב 50 זה היה מאוד מתותמם ולא עושה ניבוי טוב בכלל, וכאשר נסיתי 150 גם נתן לחלק תוצאות מדויקות ולחלק משהוא שלא הרבה תלוי אז בחרתי לוותר על הדיוק הזה ולקבל משהו שהוא עובד לכל דבר

Window : זה מגדיר את הטווח לימילה (מספר מילים מימין ושמאל) כך שהן משפעות על הווקטור שלה

אז נסיתי גם פו בהתחלה להסתכל על windows שהם יותר גדולים אבל זה לא נתן תוצאות מדויקות וגם זה הגיוני כי : מה הסיכוי שמילה מחוץ לטווח 5 היא תעזור, בדרך כלל היא לא תלויה , וגם נסיתי ערכים קטנים יותר אבל היה קצת קשה על המודל לגלות יחסים לכן נשארתי עם 5

Min_count : זה מגדיר מספר הופעות מינמלי למילה בקורפוס כדי שיהיה לה אפשריות להיות באימון

אני חושב שגודל 1 זה טוב כי ,בלי כלום הקורפוס שלנו לא גדול יחסית אז אם נחילט להתעלם ממילים שלא הופיעו הרבה זה לא החלטה טובה.

אחרי נסיון להגדיל את המספר הזה ,זה לא עזר לי לכן נשארתי עם 1 וזה פשוט יכלול כל הקורפוס

שאלות:

1.הסבירו מה המשמעות, ומה היתרונות והחסרונות של הגדלת והקטנת גודל הוקטור

כבר דברתי על זה בבחירה אבל אני אחזור
הווקטור הוא מייצג את המילה שלנו, אם אנחנו מקטנים אותו אז אנחנו לוקחים פחות מקום פחות משאווים כדי לבטא אותו , אבל במקביל גם אנחנו מפסידים דיוק כי הווקטור שלנו קטן, לא יכולים להשתמש פו כדי לבטא את הכל אין לנו מספיק מקומות לשמור כל המאפיינים ובחירה בווקטור מאוד קטן זה גורר ל under_fitting כלומר לא לומדים ומתאמנים מספיק טוב

לעומת זאת בחירת ווקטור גדול היא צריכה יותר משאווים ייותר מקום וגם בוודאי יותר זמן אבל במקביל היא נותנת לנו יותר דיווק תופס יותר מאפיינים כי היא גדול יש הרבה מקומות לשמור מאפיינים

אבל לפעמים בחירת ווקטור מאוד גדול יכולה לגרור ל over_fitting משמעות של זה שהמודל שלנו התאמן מאוד טוב על ה data שלנו אבל לעומת זאת אם נתונים לו משהו חדש אז הוא לא מסתדר טוב

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.

אנחנו חושבים שיש כמה בעיות

בעיה ראשונה היא שמודל זה אומן על משפטים וועדות של חברי הכנסת בישראל אז הוא לומד דברים פוליטיים של ישראל ומתאמן עליהם, אבל אם אנחנו רצינו להשתמש במודל הזה במדינה אחרת ,זה לא יתן תוצאות מדויקות כי יש הרבה שינוויים בצורת הטקט,משמעות ... בין מדינה למדינה אחרת

בעיה שנייה: היא פשוט שזה מודל מאומן כל courps של הכנסת אז הוא מאומן על דברים ספציבים ואז אם ננסה להשתמש פו במקומות אחרות למשל משהו תלוי באונבירסטה יכול להיות שהוא יתן תוצאות לא קרובות למה שאנחננו מדברים לו כי טווח המילים והביטויים שלו מאוד מגובל

דבר שלישי הוא שלא התייחסנו להרבה דברים שקיימים בעברית וצריך לתת להן חשיבות למשל :המילה "היקר" היא לא מילה יחידה היא מורכבת מ : "ה" , "יקר ",אז אני חושב שהיינו צריכים לתת חשיבות לדברים כאלה

דבר רביעי :

גם צריך לקחת בחשבון שקורפוס שלנו הוא ייחסית מאוד קטן לכן אני מאמין שאם נשתמש פו במקום אחר זה יהיה קצת מתומתם , כי בפועל הטקסטים הנמצאים היום הם לפעמים יהיו יותר ממילין שורה אז מה המשמעות שלהשתמש במודל שאומן על מאה אלף שורות זה לא הגיוני בכלל וזה לא יהיה יעיל אתה צריך מודל שמקסה את רוב הדברים של השימיוש שלך וזה לא מתקיים פו.

שלב 2)

סעיף א)

פונקציה זו מקבלת את המילים מ main (ישראל,כנסת ,ממשלה ,חבר...) ואז היא לכל אחת מהן עוברת על כל המילים בקורפוס חוץ מעצמה ומוצא הדמיון על ידי (mode.vw.similarty()) ואז פשוט מחזירה ה5 הכי טובות

```
def find_simliar_words(model, wantd_words, num_of_words=5):
    try:
        similar_words = {}
        voc = model.wv.key_to_index.keys()   #get all words in voc

        for wanted_word in wantd_words:
            list_of_similar_words = []
            for other_word in voc:
                if other_word != wanted_word:
                    similarity_level = model.wv.similarity(wanted_word, other_word) #calc simliarty
                    list_of_similar_words.append((other_word, similarity_level))

            list_of_similar_words.sort(key=lambda x: x[1], reverse=True)
            similar_words[wanted_word] = list_of_similar_words[:num_of_words] # get the most simliar

        return similar_words
    except Exception as e:
        print(f"An error occurred in find_simliar_words: {e}")
```

פונקציה זאת פשוט שומרת את הפלט של פונקציה קודמת בצורה הדרושה ב קובץ הפלט

```
def save_similar_words_to_file(list_of_similar_words, out_dir):
    try:
        os.makedirs(out_dir, exist_ok=True)
        out_file_path = os.path.join(out_dir, 'knesset_similar_words.txt')

        with open(out_file_path, 'w', encoding='utf-8') as f:
            for word, similar_words in list_of_similar_words.items():  #go over each word
                similar_words_list = []
                for curr_word, curr_score in similar_words:
                    word_with_level = f"({curr_word}, {curr_score})"
                    similar_words_list.append(word_with_level)

                similar_words_str = ', '.join(similar_words_list)
                curr_line = f"{word}: {similar_words_str}"
                curr_line+="\n"
                f.write(curr_line)
    except Exception as e:
        print(f"An error occurred in save_similar_words_to_file: {e}")
```

סעיף 2)

פונקציה זאת מייצרת ווקטורים כנרדש היא משתמשת בpreprare_one_sent מחלק 1 כדי להתעלם מסימני פסוק מספרים וככה ואז לכל משפט סוכמת את הווקטור שלו ובסוף מחלקים במספר המילים במשפט וככה עושים לכל משפט

```
#part 2.2
2 usages
def embedding_sentnces(model,sentnces):
    try:
        avg_sentnces=[]
        for sent in sentnces:
            tkns=prepare_one_sent(sent)
            size_of_vector=model.vector_size
            avg_vector=np.zeros(size_of_vector)
            for token in tkns:
                avg_vector += model.wv[token]
            if (len(tkns)>0):        #
                avg_vector /=len(tkns)
            avg_sentnces.append(avg_vector)
        return avg_sentnces

    except Exception as e:
        print(f"An error occurred in embedding_sentnces: {e}")
```

סעיף ג)

לגבי הערה ראשונה כבר הקורפוס שלנו מפריד את הטוקנים ברווחים לכן לא הצרכתי לטבל בזה

זאת הפונקציה שהיא מוצאת משפטים דומים פשוט מקבלת את כל המשפטים והמודל

אחר כך מגדיר את המשפטים שלי(אם מסתכלים עליהם רואים שיש רווח בין כל שנו פסוקים)

אז אחר כך אני משתמש במה שעשיתי בסעיף קודם מייצר וקטורים למשפטים שלי וגם לכל המשפטים בקורפוס ואז משתמש ב (cosine_similarty) כדי לחשב את הדמיון בין כל אחד מהמשפטים שלי לכל אחד מהשפטים בקורפוס ואז אני לוקח את המשפט השני הכי דומה כי הראשון הוא המשפט עצמו

(פו שמתי לב גם שיש משפטים שמופעים יותר מפעם אחת אז גם אם לוקחים את השני הכי דומה זה יהיה גם המשפט עצמו לכן השתדלתי לא לקחת משפט שמופיע יותר מפעם אחת)

```
def find_simliar_sentncenes(sentnces,model,out_dir):
    try:
        os.makedirs(out_dir, exist_ok=True)
        out_file_path = os.path.join(out_dir, 'knesset_similar_sentences.txt')
        chosen_sentnces=[". חבר הכנסת הרצוג , בבקשה",
                         ". חנושא ייבדק מחדש במהלך המחצית הראשונה של כהונת הכנסת",
                         ". היה דיון קודם על זה",
                         ". אני מבקש שיצביעו על ההסתייגות שלי",
                         "? מדובר על 120 אלף שקלים",
                         ". אני אוהב לסמן אותו באדום",
                         ". ברוך השם , יש לי זמן",
                         ". אנחנו עושים חזרות לקראת השבוע הבא",
                         ". עוד הפעם לא הבנת את מה שאמרתי",
                         ". אני רוצה לתת כאן נתון"]
        embeddings_courpus=embedding_sentnces(model, sentnces)
        embd_chosen_sentnces = embedding_sentnces(model,chosen_sentnces)

        similarity_mat = cosine_similarity(embd_chosen_sentnces,embeddings_courpus)
        out_line=""
```

```
        for idx,curr_sent in enumerate (embd_chosen_sentnces):
            out_line+=chosen_sentnces[idx]
            out_line+=': most similar sentence: '
            seconed_similar_sent_idx = similarity_mat[idx].argsort()[-2]  #the first one is the sentnce i
            out_line+=sentnces[seconed_similar_sent_idx]
            out_line+='\n'

        with open(out_file_path, 'w', encoding='utf-8') as f:
                f.write(out_line)

    except Exception as e:
```

סעיף ד)

פו פשוט בהתחלה מקבלים את ה model ומגדיר את המשפטים שתי פעמים ,פעם עם כוביות (מילים רוצים להחילף) ופעם בלי(עשיתי את זה כדי לזהות את המילים בצורה יותר נוחה גם אם היה מילה שאני רוצה לנבא שהיא מופיעה פעמים אז לא לקחת את הלא נכונה)

ואז אני מתחיל לעבור על המשפטים שלי (שיש להם כוכב)

ובודק כל מילה שאני רוצה להחילף ידנית (זה בשביל שאוכל לשנות את הpositive,negative,top)

```
def replace_red_words(model,out_dir):
    try:
        os.makedirs(out_dir, exist_ok=True)
        out_file_path = os.path.join(out_dir, 'sentences_words_red.txt')
        red_sentences = [
            ". *לדיון* חכנסו בבקשה , ברוכים חבאים",
            ". בתור יושבת ראש תוועדה , אני *מוכנה* להאריך את *החסכם* באותם תנאים",
            ". בוקר *טוב* , אני *פותח* את חישיבה",
            ". אנחנו שמחים *להודיע* שחברינו ח*יקר* קיבל קידום , *שלום*"
        ]

        original_sentences = [
            ". ברוכים חבאים , חכנסו בבקשה לדיון",
            ". בתור יושבת ראש תוועדה , אני מוכנה להאריך את החסכם באותם תנאים",
            ". בוקר טוב , אני פותח את חישיבה",
            ". אנחנו שמחים להודיע שחברינו תיקר קיבל קידום , שלום"
        ]
        new_sentences = []
        replaced_red_words = []

        for sent in red_sentences:
            replaced_sentence = sent
            replace_sents = []
```

לגבי מה שיצא לי על המודל שאמנתי :

נשים לב שיש תנאי שאומר אם ה train_flag=0 משמיות של זה שלא מאמנים את המודל מהאפס

ואז לקחתי את המילים ידנית הדפסתי אותם ואחר כך שמתי תוצאות באופן ידני (זה לא המקרה כאשר את תריצי את הקוד)

נתחיל ב "לדיון" :

פו לא היה עוזר לי להגדיל את ה topn וגם לא עזר לי להוסיף מילים ל negative ולא עזר לי להוסיף רק את המילה ל "לדייון" לתוך postive אבל עזר להוסיף הרבה מילים ואז בסוף סמתי 5 מילים כדי להגיע ל "למליאה" וכל 5 המילים הגיונים לסים אותם במשפט...אני חושב שחילוף זה מוצלח

עכשיוו "מוכנה" :

פו השתמשתי ב topn=3 וגם סמתי "איש" בתוך ה negative כדי שלא אקבל דברים תלויים בזכר כמו יכול או מוכן וגם הוספתי "אישה " ל 11 ositive לאותה סיבה, הוספתי גם רוצה וגם מילה עצמה ואז בסוף קבלתי "מבקשת,מנסה,חייבת" אז נסים לב שקולם לנקבה בגלל "איש" ו"אישה " והם קרובים במשמעות ל "מוכנה", ואז בסוף בחרתי ב "מנסה" , אני חושב שחילוף זה היה מוצלח

כעת "ההסכם":

פו זה קצת לא היה קל נסיתי הרבה אפשריות והרבה מילים ואפילו מילים שהם מאוד קרובים במשמעות ל"ההסכם" כמו "החוזה" אבל לא נתן משהו טוב אבל בסוף השתמשתי ב topn=3 עם מילים ב postive שהם "ההסכמה","האמנה" ואז קבלתי את "ההתחייבות " זאת לא מתאימה , "התופעה " גם זאת לא מתאימה והשלישית היא "המשימה" שהיא כן טובה ומותאמת ויכולה להחליף "ההסכם"

```
replace_sents = []
if(train_flag==0):
    if '*לדיון*' in sent:  #  top_n 3 פו קבלתי "למליאה" במקום לדיון אז אין צורך ב
        debate = model.wv.most_similar(positive=['לדיון','למשלה', 'לוועדה','ליישיבה','לכנסת'], negative=[], topn=1)[0][0]
        #print("debate=",debate)
        replaced_sentence = replaced_sentence.replace( _old: '*לדיון*', debate)
        replace_sents.append(('לדיון', debate))
    if '*מוכנת*' in sent:
        ready = model.wv.most_similar(positive=['רוצה', 'מוכנה', 'אישה'], negative=['איש'], topn=3)
        #print("ready=",ready)
        #  קבלתי[מבקשת,מנסה,חייבת]
        replaced_sentence = replaced_sentence.replace( _old: '*מוכנת*',  _new: 'מנסה')
        replace_sents.append(('מוכנה', 'מנסה'))
    if '*ההסכם*' in sent: #              קבלתי "המעשה"וזה מספיק טוב
        agreement = model.wv.most_similar(positive=['האמנה','ההסכמה'], negative=[], topn=3)
        #print("agreement=",agreement)
        #          [ההתחייבות,התופעה,המשימה] קבלתי
        replaced_sentence = replaced_sentence.replace( _old: '*ההסכם*',  _new: 'המשימה')
        replace_sents.append(('ההסכם', "המשימה"))
```

"טוב " :

זאת גם הייתה מילה קצת בעייתית ,בהתחלה נסיתי לסים אור ב positive אבל היה שם של מישהו שהוא גם "אור" לכן זה לא עבד, ואז התחלתי לנסות מילים דומות לטוב וזה גם לא עבד מספיק טוב, וגם נסיתי לסים את "רע" בצד ה negative אבל הפתיע אותי שהוא היה מנבא מילים דומים לרע כמו,"מזעזע","כואב",ואז בסוף סמתי "רע" ,"מצויין","נפלא " ב pos וקבלתי [נחמד,מוגזם,דומה] ברור שנחמד הכי מתאימה, אני חושב שזאת החלפה מוקבלת

"פותח":

פו נסיתי להשתמש ב "מתחיל" , "ממשיך " אבל לא עבד ואז בסוף החלטתי להשתמש בtopn=3 עם "פותח","מעביר"

וקבלתי :[מתחחיב,מעלה,מעדיף] והכי מתאימה היא מעלה , אני חושב שזאת החלפה מוצלחת

"שלום": פו פשוט השתמשתי בכמה מילים דמויים ל "שלום" שהן "היי" , "רבותי" ו "שלום" עצמה ,

הניבוי זה לא היה מאוד קשה והשמתמשי גם ב topn=3 וקבלתי "רבותיי" שלדעתי ניבוי זה מאוד מוצלח

```
if '*טוב*' in sent:
    good = model.wv.most_similar(positive=['רע', 'מצויין', "נפלא"], negative=[], topn=3)
            #  (negative=[], topn=3 , [ 'טוב', 'מצויין', 'רע']=positive)good = model.wv.most_similar
    # Z[דומה , מוגזם , נחמד] קבלתי
    #print("good=",good)
    replaced_sentence = replaced_sentence.replace( _old: '*טוב*',  _new: 'נחמד')
    replace_sents.append(('טוב', "נחמד"))

if '*פותח*' in sent:
    _open = model.wv.most_similar(positive=[ "מעביר" ,'פותח'], negative=[], topn=3)
    # [מעדיף,מעלה,מתחייב] קבלתי
    #print("open=",_open)
    replaced_sentence = replaced_sentence.replace( _old: '*פותח*',  _new: "מעלה")
    replace_sents.append(('פותח', "מעלה"))

if '*שלום*' in sent:
    hi = model.wv.most_similar(positive=[ 'היי', 'רבותי' ,'שלום'], negative=[], topn=3)
    #       [בכם,מחליכוך,רבותיי]קבלתי
    #print("hi=",hi)
    replaced_sentence = replaced_sentence.replace( _old: '*שלום*',  _new: "רבותיי")
    replace_sents.append(('שלום', "רבותיי"))
```

"להודיע":

פו זה לא היה מאוד קשה נסיתי קודם ב "להודיע" , "להזכיר "

לא הצלתחתי אחרי כמה נסיוניות נסיתי "להגיד" עם "לספר", עם topn=3

וקבלתי: [להסביר,לבשר ,להזכיר] ואני חושב ש "לבשר "הכי מתאימה, חושב זה ניבוי מוצלח

"יקר":

פו נסיתי כמה דברים למשל "מצויין","טוב" אבל לא קבלתי משהו ואז השתמשתי ב topn=3 עם המילים

"יקר","יפה","טוב" ואז קבלתי :[חזק ,רע ,בריא] ואז אני חושב ש "חזק " היא הכי מתאימה פו,ניבוי זה מקובל

```
if '*להודיע*' in sent:
    tell = model.wv.most_similar(positive=['לספר', 'להגיד'], negative=[], topn=3)
    #                 [להסביר,לבשר,להזכיר] קבלתי
    #print("tell=",tell)
    replaced_sentence = replaced_sentence.replace( _old: '*להודיע*',  _new: "לבשר")
    replace_sents.append(('להודיע', "לבשר"))

if '*יקר*' in sent:################
    #(negative=[], topn=3 , ['אור', 'טוב', 'מצויין', 'רע']=positive)good = model.wv.most_similar
    val_ = model.wv.most_similar(positive=['טוב','יפה','יקר'], negative=[], topn=3)
    #         [חזק,רע,בריא] קבלנו
   # print("value=",val_)
    replaced_sentence = replaced_sentence.replace( _old: '*יקר*',  _new: "חזק")
    replace_sents.append(('יקר', "חזק"))

new_sentences.append(replaced_sentence)
replaced_red_words.append(replace_sents)
```

פו אן ה train_flag=1 משמעות של זה אנחננו מאמינם את המודל שלנו מאפס ואז תמיד לקחת המילה במקום ה[0] [0])השתמשתי באותם pos,neg כמו קודם(,

זה המקרה כשאת מריצה את הקוד שלנו (כלומר מוצאים מילות חדשות מההתחלה על המודל החדש)

```
else:
    if '*לדיון*' in sent:
        debate = model.wv.most_similar(positive=['לדיון', 'למחשלה', 'לוועדה', 'לישיבה', 'לכנסת'], negative=[],
                                       topn=1)[0][0]
        replaced_sentence = replaced_sentence.replace( _old: '*לדיון*', debate)
        replace_sents.append(('לדיון', debate))
    if '*מוכנה*' in sent:
        ready = model.wv.most_similar(positive=['רוצה', 'מוכנה', 'אישה'], negative=['איש'], topn=3)[0][0]
        replaced_sentence = replaced_sentence.replace( _old: '*מוכנה*', ready)
        replace_sents.append(('מוכנה', ready))
    if '*הסכם*' in sent:
        agreement = model.wv.most_similar(positive=['האמנה', 'הסכמה'], negative=[], topn=3)[0][0]
        replaced_sentence = replaced_sentence.replace( _old: '*הסכם*', agreement)
        replace_sents.append(('הסכם',agreement))
    if '*טוב*' in sent:
        good = model.wv.most_similar(positive=['רע', 'מצוייו', "נפלא"], negative=[], topn=3)[0][0]
        replaced_sentence = replaced_sentence.replace( _old: '*טוב*', good)
        replace_sents.append(('טוב', good))
    if '*פותח*' in sent:
        _open = model.wv.most_similar(positive=["מעביר", 'פותח'], negative=[], topn=3)[0][0]
        replaced_sentence = replaced_sentence.replace( _old: '*פותח*', _open)
        replace_sents.append(('פותח', _open))
```

```
if '*שלום*' in sent:
    hi = model.wv.most_similar(positive=['היי', 'רבותי', 'שלום'], negative=[], topn=3)[0][0]
    replaced_sentence = replaced_sentence.replace( _old: '*שלום*', hi)
    replace_sents.append(('שלום', hi))

if '*להודיע*' in sent:
    tell = model.wv.most_similar(positive=['לספר', 'להגיד'], negative=[], topn=3)[0][0]
    replaced_sentence = replaced_sentence.replace( _old: '*להודיע*', tell)
    replace_sents.append(('להודיע', tell))

if '*יקר*' in sent:
    val_ = model.wv.most_similar(positive=['טוב', 'יפה', 'יקר'], negative=[], topn=3)[0][0]
    replaced_sentence = replaced_sentence.replace( _old: '*יקר*', val_)
    replace_sents.append(('יקר', val_))
```

זה פשוט לכתוה תוצאות לקובץ

```
    with open(out_file_path, 'w', encoding='utf-8') as file:
        for original, changed, replace_sents in zip(original_sentences, new_sentences, replaced_red_words):
            file.write(f"{original}: {changed}")
            file.write("\n")
            replaces_strs = []
            for old, new in replace_sents:
                rep_str = f"({old}:{new})"
                replaces_strs.append(rep_str)

            replace_sents_str = ', '.join(replaces_strs)
            file.write(f"Replaced words: {replace_sents_str}")
            file.write("\n\n")

except Exception as e:
    print(f"An error occurred in replace_red_words: {e}")
```

ואז התוצאות הן

```
ברוכים הבאים , הכנסו בבקשה לדיון .: ברוכים הבאים , הכנסו בבקשה למליאה .
Replaced words : (לדיון:למליאה)

בתור יושבת ראש הוועדה , אני מוכנה להאריך את ההסכם באותם תנאים .: בתור יושבת ראש הוועדה , אני מנסה להאריך את המשימה באותם תנאים .
Replaced words : (מוכנה:מנסה), (ההסכם:המשימה)

בוקר טוב , אני פותח את הישיבה .: בוקר נחמד , אני מעלה את הישיבה .
Replaced words : (טוב:נחמד), (פותח:מעלה)

שלום , אנחנו שמחים להודיע שחברינו היקר קיבל קידום .: רבותיי , אנחנו שמחים לבשר שחברינו החזק קיבל קידום .
Replaced words : (שלום:רבותיי), (להודיע:לבשר), (יקר:חזק)
```

אני חושב שכן הצלחתי במשימה

רק בוקר נחמד זה קצת לא הכי יפה בעברית אבל מקובל

שאלות:

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.

```
ישראל: (שיגדירו, 0.7952374219894409), (אריסון, 0.7922749519348145), (בחיזוקה, 0.78355872631073), (היצואן, 0.7690911889076233), (וחששנו, 0.7666418552398682)
כנסת: (ולידיעת, 0.7647874355316162), (וחבריי, 0.7643329501152039), (מפלגתו, 0.7291432619094849), (שזכויותיהם, 0.7267552614212036), (במרביתן, 0.7261114716529846)
ממשלה: (טגנו, 0.8995431661605835), (עיר, 0.8911945819854736), (העיר, 0.846303403377533), (הממשלה, 0.8308998346328735), (לאפידמולוגיה, 0.8252911567687988)
חבר: (לחבר, 0.8421520590782166), (שחבר, 0.8420298099517822), (וחבר, 0.8293347358703613), (חברת, 0.8076171278953552), (מזכיר, 0.791201651096344)
שלום: (מתיישב, 0.8111182451248169), (משא-ומתן, 0.8069940209388733), (הסכם, 0.8064852952957153), (ערבי, 0.8059404492378235), (ערפאת, 0.8025083541870117)
שולחן: (כהונת, 0.8598305583000183), (הודעת, 0.8389068245887756), (תקנון, 0.8342778086662292), (בנשיאות, 0.8306841254234314), (להכרעת, 0.8304879665374756)
מותר: (אסור, 0.9178904891014099), (אכפת, 0.8909635543823242), (נתת, 0.8836268782615662), (כדאי, 0.8756571412086487), (תנו, 0.8751662969589233)
מדבר: (מדברת, 0.8792674541473389), (מסתכל, 0.878637969493866), (סומך, 0.8684264421463013), (חולק, 0.8667044043540955), (מסתכלת, 0.8534050583839417)
ועדה: (בהצהרה, 0.8204995393753052), (שרירותית, 0.8192328214645386), (לחקיקה, 0.8169746398925781), (לממשלה, 0.8144350051879883), (שתוקם, 0.8136600852012634)
```

אלה הן התוצאות ולגבי השאלה : לרוב לא המילים לא תואמות את הצפיות שלי

נקח למשל "ישראל" מה זה התלות בין "ישראל" ל "וחחשנו"

"ישראל" שם ו"חששנו" פועל ואין תלות בינהם

גם " שיגדירו" אינה תלויה ב "ישראל"

רק "אריסון" זה שם אז זה כן מתאים קצת

אבל יש לדוגמה "ממשלה" ו "עיר" שהן כן דומות , "ממשלה" ו "הממשלה" אותה מילה כמעט

הפתיע אותי קצת ש "מותר" ו "אסור" מאוד דומות כי הן אחת ההפך של השני

למה זה קרה פשוט כי יש לנו קורפוס מאוד ספתיבי וקטן אבל אם למשל הייה לנו קורפוס מכיל דיבוריים של כל הממשלות בעולם , אז יהיה יותר טוב כי למשל "ישראל" יכולה עכשיוו להיות מותאמת לשם מדינה אחר כי יש הרבה שמות מדינות עכשיו,אז לרוב אני חושב בגלל גודל ה קורפוס (גם שהוא לא

מכיל הרבה דברים שונים וגם שיטת האימון אני לא רואה שהיא מאוד מותחכמת כדי לקבל משהו מאוד טוב יש שיטות יותר טובות.

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? הסבירו.

בתור רעיון התחלתי ,אז לא חשבתי שבין מיליות antonyms יהיה מרחק קטן אלה ההפך

כי בסופו של דבר אלה מילים הפוכות והווקטור מציג את המילה וכשכל ששני ווקטורים הם קרובים אז המילים שלהם גם קרובות ואז לא הגיוני שמילה וההפך שלה קרובות.

אחרי מה שקרה עמי בסעיף 2.1 שקבלתי ש "מותר" ו "אסור" דומות

חשבתי קצת וגליתי שכן זה הגיוני כי הרבה פעמים אנחנו אומרים את המילה וההפך שלה באותה משפט ואז אם היא בתוך הwindow היא תכלול בווקטור של המילה

למשל "לשבת מותר אך אסור לעשות מנגל" ואז מותר תכלול בווקטור של אסור וגם ההפך

3)מצאו שלושה זוג ות של מילים שנחשבות להפכים(antonyms(הקיימות בקורפוס שלנו ובדקו את המרחק ביניהן. האם הציפייה שלכם מסעיף 2 מתקיימת עבורן עם המודל שבניתם?

עשיתי יותר מ 3 כדי שלא יהיו מקרים ספציבים

(זה לא דמיון זה המרחק(כמו שדרוש בשאלה(המרחק)))

```
(0.427518963813781 74 :בוקר , ליל)
(0.20757102966308594 :קשה , קל)
(0.23195225000381 47 :טוב , רע)
(0.19981688261032104 :קשוח , יפה)
(0.292344331741333 :ארץ , שמים)
(0.08210951089859009 :אסור , מותר)
(0.04977762699127197 :שנאה , אהבה)
```

אז אם אנחנו מסתכלים ,אנחנו רואים שהמרחקים קטנים יחסית בין כל הזוגות

אפילו "אהבה" , "שנאה" הן מאוד קרובות

המיליות הכי רחוקות הן "ליל" , "בוקר" וגם הם לא מאוד רחוקים

כן זה מתאים לצפיות שלי אבל יותר מדי כאילו לקחתי 7 דוגמאות וכולן יחסית המרחק בינהם קצר.

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.

האמת שצפיתי לתוצאות מאוד גרועות אבל נפתעתי שהן יחסית מוקבלות,כאילו לא חשבתי שהשיטה לקחת את הממוצע של מילים במשפט בלי להתייחסות לסדר בלי כלום זה יגרור למשהו טוב,אבל אחרי חשיבה אז זה יכול להיות שהוא עבד יחסית טוב כי הקורפוס שלנו קטן ועל מקצוע מיוחד ואז כל המשפטים שיש בהם אותם מילים הם יהיו על אותו מקצוע כי אין הרבה שורות ודברים כאלה

אבל אם נקח קורפוס גדול זה לא כל כך יעבוד טוב

```
חבר הכנסת הרצוג , בבקשה .: most similar sentence: חבר הכנסת בועז ביסמוט בבקשה .
הנושא של הזרקת מים לבשר הקפוא עלה על שולחן הכנסת בקדנציה הקודמת , וועדת הכלכלה של הכנסת .: most similar sentence: הנושא ייבדק מחדש במהלך המחצית הראשונה של כהונת הכנסת
דנה בו ארוכות .
היה דיון קודם על זה .: most similar sentence: בוא נעשה דיון על זה .
אני מבקש שיצביעו על ההסתייגות שלי .: most similar sentence: אני מבקש להצביע על ההצעה שלי .
מדובר על 120 אלף שקלים ?: most similar sentence: מדובר על 800 מיליון שקל לסגירת הפער .
אני אוהב לסמן אותו באדום .: most similar sentence: אני הוצאתי אותו החוצה .
ברוך השם , יש לי זמן .: most similar sentence: לי יש זמן .
אנחנו עושים חזרות לקראת השבוע הבא .: most similar sentence: אנחנו נעשה טעימה קטנה בסיבוב הבא .
עוד הפעם לא הבנת את מה שאמרתי .: most similar sentence: לא שמעו את מה שאמרתי , רגע .
אני רוצה לתת כאן נתון .: most similar sentence: אני רוצה להגיד כאן בפה מלא :
```

אם נסתכל על המשפט הראשון זה מאוד דומה

חבר הכנסת הרצוג , בבקשה . חבר הכנסת בועז ביסמוט בבקשה .

יש גם משפט 3

היה דיון קודם על זה . בוא נעשה דיון על זה .

יחסית כולהן קרובות

שלב 3)

זאת פונקציה שמקבלת את ה data ואת גודל ה chunk ומחזירה את ה chunks (כבר ממשנו אותה מתרגיל 3)

```python
def get_Chunks(df, chunk_size=1):
    combined_rows = []

    committee_group = df[df['protocol_type'] == 'committee']
    committee_sentences = committee_group['sentence_text'].tolist()
    num_committee_chunks = len(committee_sentences) // chunk_size
    for i in range(num_committee_chunks):
        start_idx = i * chunk_size
        end_idx = (i + 1) * chunk_size
        chunk_sentences = ' '.join(committee_sentences[start_idx:end_idx])
        knesset_number = committee_group.iloc[start_idx]['knesset_number']
        combined_rows.append({'protocol_type': 'committee', 'sentence_text': chunk_sentences, 'knesset_number': knesset_number})

    plenary_group = df[df['protocol_type'] == 'plenary']
    plenary_sentences = plenary_group['sentence_text'].tolist()
    num_plenary_chunks = len(plenary_sentences) // chunk_size
    for i in range(num_plenary_chunks):
        start_idx = i * chunk_size
        end_idx = (i + 1) * chunk_size
        chunk_sentences = ' '.join(plenary_sentences[start_idx:end_idx])
        knesset_number = plenary_group.iloc[start_idx]['knesset_number']
        combined_rows.append({'protocol_type': 'plenary', 'sentence_text': chunk_sentences, 'knesset_number': knesset_number})

    return pd.DataFrame(combined_rows)
```

עכשיו פו עושים שלב האימון ,מתחילים לעבור על כל הגדלים של ה chunks

-משתמשים ב get_chunks בשביל ליצר אוצם

-עושים איזון ל data (כמות הplenary_chunks=commitme_chunks)

מחשבים את ה embedding_vectors לכל המשפטים בשימוש בפונקציה שהגדרנו בחלק 2

```python
1 usage
def class_method(model, corpus_data, chunks_sizes=[1, 3, 5], num_of_neighbors=5):
    for curr_chunk_size in chunks_sizes:
        print(f"chunk size: {curr_chunk_size}")
        print(f"num_of_neighbors: {num_of_neighbors}")

        # build chunks
        corpus_chunks = get_Chunks(corpus_data, curr_chunk_size)

        #balance the ckunks
        committee_indexes = corpus_chunks[corpus_chunks['protocol_type'] == 'committee'].index
        plenary_indexes = corpus_chunks[corpus_chunks['protocol_type'] == 'plenary'].index
        down_sampled_indexes = np.random.choice(plenary_indexes, size=len(committee_indexes), replace=False)
        combined_indexes = np.concatenate([committee_indexes, down_sampled_indexes])
        corpus_chunks = corpus_chunks.loc[combined_indexes].reset_index(drop=True)

        labels= corpus_chunks['protocol_type']

        embeddings = embeddings_sentence(model, corpus_chunks['sentence_text'])
```

-מחלקים את ה data ל test,train

-בונים את ה knn ומאמנים אותו

- ואז מדפסים את ה classification_report

```
# split  data to train and test
X_train, X_test, y_train, y_test = train_test_split( *arrays: embeddings, labels, test_size=0.1, random_state=42, stratify=labels)

# train knn
knn = KNeighborsClassifier(n_neighbors=num_of_neighbors)
knn.fit(X_train, y_train)

# predict and evaluate
y_pred = knn.predict(X_test)

print(classification_report(y_test, y_pred))
```

ב main אני פשוט קורא לפונקציה

אלה הן תוצאות של ה classifaction_report

```
chunk size: 3
num_of_neighbors: 5
              precision    recall  f1-score   support

   committee       0.67      0.69      0.68      1391
     plenary       0.68      0.66      0.67      1390

    accuracy                           0.68      2781
   macro avg       0.68      0.68      0.68      2781
weighted avg       0.68      0.68      0.68      2781
```

```
chunk size: 1
num_of_neighbors: 5
              precision    recall  f1-score   support

   committee       0.62      0.65      0.63      4171
     plenary       0.63      0.60      0.61      4171

    accuracy                           0.63      8342
   macro avg       0.63      0.63      0.62      8342
weighted avg       0.63      0.63      0.62      8342
```

```
chunk size: 5
num_of_neighbors: 5
              precision    recall  f1-score   support

   committee       0.69      0.70      0.70       835
     plenary       0.70      0.68      0.69       834

    accuracy                           0.69      1669
   macro avg       0.69      0.69      0.69      1669
weighted avg       0.69      0.69      0.69      1669
```

שאלות:

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?

משמאל מופיע תוצאה קודמת , כך שווקטור המאפינים היה tfidf ואמננו על ידי שיטת החלוקה עם מספר שכנים =5 עם chunk_size שווה ל 5 ומימן אותו דבר רק עם הווקטור החדש, אז ברור קיבלנו תוצאות פחות טובות ממה שקיבלנו קודם

```
KNN with split:
              precision    recall  f1-score   support

   committee       0.84      0.85      0.85       835
     plenary       0.85      0.84      0.84       834

    accuracy                           0.85      1669
   macro avg       0.85      0.85      0.85      1669
weighted avg       0.85      0.85      0.85      1669
```

```
chunk size: 5
num_of_neighbors: 5
              precision    recall  f1-score   support

   committee       0.69      0.70      0.70       835
     plenary       0.70      0.68      0.69       834

    accuracy                           0.69      1669
   macro avg       0.69      0.69      0.69      1669
weighted avg       0.69      0.69      0.69      1669
```

2. עבור התשובה שעניתם בסעיף ,1 הסבירו מדוע לדעתכם זה קרה.

אני חושב שזה קורה בגלל מספר התכונות ,קודם היה לנו ב tf_idf שכל מילה מקבלת איזה שהוא משקל ספתסיבי משלה וגם הוא מנסה לתפוס תלות בין מילים, וזה יותר מדויק ,אבל עכשיוו יש לנו רק 100 מקומות שאנחנו רוצים להציג את המילים שלנו על ידיהן ואז זה נותון משמעות כללית למילה שלנו ,אז הווקטור לי ייצג אותה בצורה מדויקת , לכן קודם קיבלנו תוצאות יותר טובות.

3 עבור איזה גודל צ'אנק קיבלתם תוצאות יותר טובות? האם זה נכון גם לגבי וקטורי המאפיינים שהשתמשתם בהם בתרגיל 3? הסבירו.

קבלנו את התוצאות הכי יפות עבור chunk בגודל 5 , כן זה גם היה נכון עבור תרגיל קודם קבלנו ש צ,אנק בגודל 5 יותר טוב מ (1,3)

הסיבה לזה היא פשוט ש chunk בגודל 5 יש לו יותר משפטים ולכל משפט יש מידע ולאחד אותם זה מוסיף את מספר המידע של ה chunk ,ואז הוא לא יהיה מורכז במידע של משפט יחיד,אלה בהרבה מידע ואז הוא יכול לתפוס קשרים יותר רחבים ,וגם יהיה פחות רעיש

ולהפך כאשר ה chunk הוא 1 או 3 אז זה תופס פחות מידע פחות קשרים לא מרחיב את ההסתכלות שלו ומפסיד הרבה קשרים.

וגם להזכיר שתאנק בגודל 1 יותר גרוע מ3 בשתי התרגילים

שלב 4)

בהתחלה אנחנו עושים load ל model וה toknizer

```
tokenizer = AutoTokenizer.from_pretrained('dicta-il/dictabert')  #load toknizer
model = AutoModelForMaskedLM.from_pretrained('dicta-il/dictabert')
model.eval()


print_prediction = ''
os.makedirs(out_dir, exist_ok=True)
out_file_path = os.path.join(out_dir, 'dictabert_results.txt')
```

פונקיצה זו מקבלת המודל ,הטוקניזר והמשפט (שמכיל באיזה שהוא מקום [MASK]) ואז פונקציה זאת מחליפה אותה במילה חדשה ,

-פשוט משתמשים בטוקניזר קודם כדי לקבל encode ווקטור ,אחר כך מעבירים אותם במודל

ואז מוצאים את הטוקן המוחלף ואז מחזירים אותו עם משפט מעודכן

```
def replace_masked_token(model, tokenizer, sent):
    try:

        # make encode to the sent and find the index of  [MASK]
        input_ids = tokenizer.encode(sent, return_tensors='pt')
        # find our Mask tkn
        mask_token_index = torch.where(input_ids == tokenizer.mask_token_id)[1]
        with torch.no_grad(): #we learned in deep learining that we should put the forward in this

            outputs = model(input_ids)
        # take  the logits
        sentence_logits = outputs.logits[0]
        mask_token_logits = sentence_logits[mask_token_index, :].squeeze()

        # get  the predicted token
        top_token_index = torch.argmax(mask_token_logits, dim=-1)
        top_token = top_token_index.item()

        # get the new token and updated sent
        pred_token = tokenizer.decode([top_token]).strip()
        solved_sent = sent.replace(tokenizer.mask_token, pred_token, 1)
        return solved_sent, pred_token
    except Exception as e:
        print(f'Exception at replace_masked_token: {e}')
```

אז פו אנחנו מתחילים לעבור על המשפטים וכל פעם מחליפים את ה * הבאה ב MASK כי ככה dictabert עובד צריך שיהיה בסנטקס הזה, לאחר מכן קוראים לפונקציה שעושה חילוף, ומסדרים את הפלט כמו שצריך

```
out_file_path = os.path.join(out_dir, 'dictabert_results.txt')

with open(mask_file_path, 'r', encoding='utf-8') as file:
    for line in file:
        pred_tkn_list = []
        curr_sent = line.strip()

        print_prediction +="Original sentence: " +curr_sent
        print_prediction+='\n'
        num_updattes = curr_sent.count('[*]')

        for i in range(num_updattes):
            curr_sent = curr_sent.replace( __old: '[*]', __new: '[MASK]', __count: 1)
            curr_sent, predicted_token = replace_masked_token(model,tokenizer,curr_sent)
            pred_tkn_list.append(predicted_token)

        print_prediction += 'DictaBERT sentence: ' + curr_sent
        print_prediction+='\n'
        print_prediction += 'DictaBERT tokens: ' + ', '.join(pred_tkn_list)
        print_prediction+='\n\n'
```

פו פשוט מדפיסים אותו לתקייה

```
with open(out_file_path, 'w', encoding='utf-8') as output_file:
    output_file.write(print_prediction)
```

שאלות

1)האם קיבלתם משפטים הגיוניים? מבחינת התוכן, קוהרנטיות ומבחינה תחבירית. .

כן כמעט כל המשפטים הם נכונים מבחינה תחבורחית והם גם קוהרנטיות וגם משלימים את המשמעות של המשפטים, לא ש סתם זורק איזה שהוא מילה וגם אין טעיות תחבריות

זאת דוגמה

```
Original sentence: יש צורך [*] ביצירת מקומות עבודה רבים יותר , בתשלום שכר [*] בעד העבודה .
DictaBERT sentence: יש צורך גם ביצירת מקומות עבודה רבים יותר , בתשלום שכר גבוה בעד העבודה .
DictaBERT tokens: גם, גבוה

Original sentence: אבל הנושא הוא [*] אם אתה בעד [*] ההתנתקות או נגד .
DictaBERT sentence: אבל הנושא הוא לא אם אתה בעד תוכנית ההתנתקות או נגד .
DictaBERT tokens: לא, תוכנית

Original sentence: לכן , הממשלה מתנגדת [*] החוק הזאת , [*] להביא לפתרון הבעיה בדרך [*] .
DictaBERT sentence: לכן , הממשלה מתנגדת להצעת החוק הזאת , כדי להביא לפתרון הבעיה בדרך אחרת .
DictaBERT tokens: להצעת, כדי, אחרת
```

2) השוו את התוצאות שקיבלתם עכשיו לאלו שקיבלתם בתרגיל בית .2 האם יש שיפור בתוצאות לדעתכם

כן יש שיפור משמעותי, אם נזכר בתרגיל 2 אז לרוב היינו מנבים סימני פסוק,מילות מופיעות הרבה כמו [לא ,את] מילים כאלה שהם לא תמיד צריך לנבא אותם ובמקרים מיוחדים מקבלים תוצאות טובות אבל פו כמעט כל התוצאות טובות ובעלי משמעות

זאת דוגמה להשוואה:

```
Original sentence: יש צורך [*] ביצירת מקומות עבודה רבים יותר , בתשלום שכר [*] בעד העבודה .
DictaBERT sentence: יש צורך גם ביצירת מקומות עבודה רבים יותר , בתשלום שכר גבוה בעד העבודה .
DictaBERT tokens: גם, גבוה
```

```
Original sentence: יש צורך [*] ביצירת מקומות עבודה רבים יותר , בתשלום שכר [*] בעד העבודה .
Committee sentence: יש צורך , ביצירת מקומות עבודה רבים יותר , בתשלום שכר . בעד העבודה .
Committee tokens: ,,.
Probability of committee sentence in committee corpus: -131.33784555134136
Probability of committee sentence in plenary corpus: -130.25714741568706
This sentence is more likely to appear in corpus: plenary
Plenary sentence: יש צורך בהשקעה ביצירת מקומות עבודה רבים יותר , בתשלום שכר דירה בעד העבודה .
Plenary tokens: בהשקעה,דירה
Probability of plenary sentence in plenary corpus: -131.3690255729834
Probability of plenary sentence in committee corpus: -132.92538943402698
This sentence is more likely to appear in corpus: plenary
```

דוגמה 2)

```
DictaBERT tokens: ויא, כבו

Original sentence: עזבי את [*] 84 , אני לא מכיר את חוק [*] והבנייה .
DictaBERT sentence: עזבי את סעיף 84 , אני לא מכיר את חוק התכנון והבנייה .
DictaBERT tokens: סעיף, התכנון
```

```
Original sentence: עזבי את [*] 84 , אני לא מכיר את חוק [*] והבנייה .
Committee sentence: עזבי את . 84 , אני לא מכיר את חוק רשות והבנייה .
Committee tokens: .,רשות
Probability of committee sentence in committee corpus: -97.0011378074553
Probability of committee sentence in plenary corpus: -100.86125574687044
This sentence is more likely to appear in corpus: committee
Plenary sentence: עזבי את , 84 , אני לא מכיר את חוק ההסדרים והבנייה .
Plenary tokens: ,,ההסדרים
Probability of plenary sentence in plenary corpus: -97.9058078205034
Probability of plenary sentence in committee corpus: -95.75096264567506
This sentence is more likely to appear in corpus: committee
```

אז אלה רק 2 דוגמאות אבל כמעט הכל ככה לכן ברור שיש שיפור

3)האם בכל פעם שתריצו את המודל על אותו קלט תקבלו את אותו פלט? אם כן ואם לא נסו להסביר מדוע

כן אנחנו מקבלים תמיד אותה תוצאה , הסיבה לזה פשוט שאנחנו לא מתערבים במודל בשום אופן לא עושים training או pretraining לא בוחרים משתנים, לא משנים וכנראה שהמודל הוא לא הסבתרותי (הכוונה שלא הסבתרותי היא לא שהוא לא משתמש בהסבתרות,הכוונה היא שאין לו מקרים לעשות ככה בהסתברות x ואז לעשות משהו אחר בהסתברות y ואז תמיד יש סיכוי לאחד מהם) לכן הוא תמיד יש לו אותה משפטים ואני לא מתערב בפרמטרים אז יש לו אותם שלבים שהוא מבצע אותם על פעם על המשפטים שלי לכן תמיד הוא מגיע לאותו מקום בכל שלב ואז לאותם תוצאות.

4.)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.

את האמת שהוא עבד מאוד טוב , לא צפיתי את זה

יש רק את המשפט הזה שהוא נכון אבל הוא החליף את המילה האחרונה ב " ,מה שזה לא מוסיף אף משמעות למשפט ,היה יכול לנבא משהו יותר טוב , בפרט אני אומר את זה כי אין גרש אחר שסוגר את הגרש שהוא פתח ,אז זה לדעתי לא הכי טוב

```
Original sentence: בסופו של דבר , גם אני הייתי [*] אליפויות העולם רצוף שבע שנים , וכן יצאנו עם דגל ישראל ואני יודעת שחוץ מאימא שלי , אף [*] לא יודע שבאמת אנחנו מייצגים את ישראל [*] .
DictaBERT sentence: בסופו של דבר , גם אני הייתי בכל אליפויות העולם רצוף שבע שנים , וכן יצאנו עם דגל ישראל ואני יודעת שחוץ מאימא שלי , אף אחד לא יודע שבאמת אנחנו מייצגים את ישראל " .
DictaBERT tokens: בכל, אחד, "
```

גם פו קרה אותו דבר אבל למילה ראשונה

```
Original sentence: בעצם הגענו להסכמות [*] רחבות , שקשורות בעיקר לצרכים של השקיפות בכל מה שקשור בתהליך קבלת התרומות האלה .
DictaBERT sentence: " בעצם הגענו להסכמות מאוד רחבות , שקשורות בעיקר לצרכים של השקיפות בכל מה שקשור בתהליך קבלת התרומות האלה .
DictaBERT tokens: ", מאוד
```

הסיבה לכך

לדעתי יש שתי סיבות

1)אין מודל מושלם שאין פו אחוז של טעות ,זה לא אלגורתים לעשות משהו זה מודל לניביוי

2)אנחנו לא אממנו את המודל אפילו על הקורפוס שלנו לכן הוא לא יודיע את כל המאפיינים והקשרים שיש לנו ויכול להיות שהוא יתנהג מאוד טוב עבור משפטים כי הוא ראה משהו דומה אבל עבור משפטים אחרים לא כי לא היה משהו דומה בעבר