

עיבוד שפה טבעית (20225211) 2024-2025 - תרגיל 4 חלק 2 (4 נק')

שלום לכולם!

כעת נתבונן קצת יותר לעומק בשיטות ליצירת אוצרות מילים (vocabularies) מתחת לרמת המילה (subword). נאסוף קורפוס, נאמן טוקניזר לעברית שמתאים ל-BERT, ונבחן את תכונותיו.

בסיום העבודה, הגישו **קובץ pdf** הכולל את כל הפלטים מכל התאים. דרך אפשרית אחת להשיג קובץ pdf היא על-ידי הדפסת הדף (מתוך colab, לא מהדפדפן) ולבחור "מדפסת" ישירה ל-pdf.

שימו לב שכל המידע הדרוש מופיע בתדפיס ה-pdf --- לא יתקבלו ערעורים על-בסיס חיתוך. אין לחרוג ממבנה התאים ואין לענות במקומות שאינם מוגדרים לכך. אין למחוק הערות שמסמנות את מיקומי קטעי הקוד שלכם ואין לערוך קוד שנמצא מחוץ לתחומים המוגדרים. סעיפים שחורים מהנחיה זו לא ייבדקו.

ניתן להגיש בצוותים של עד שלושה סטודנטים. לא יהיו הנחות לצוותים קטנים. אפשר לחלק את פתרון התרגיל בין חברי. הצוות כרצונכם. אין להסתכל על קוד של צוותים אחרים ואין להקריא קוד לצוותים אחרים.

אין להשתמש ברכיב ג'מיני Gemini או כל סיוען מתקדם אחר.

מועד ההגשה הוא דקה לפני תחילת השיעור ב-13/1. לא יינתנו הארכות.

בהצלחה,

-- יובל

התקנות

דבר ראשון, נתקין את החבילות הנדרשות, כולן מבית האגינפייס.

```
!pip install datasets -q
!pip install tokenizers -q
!pip install transformers -q
```

נייבא חבילות. את os נצטרך כדי לכתוב את הדאטאסט שלנו לטובת האימון, וכדי לכתוב את אוצר המילים של הטוקניזר לאחר האימון כדי שיוכל להיטען לאובייקט הקורא אותו.

```
import os
import datasets
from tqdm.notebook import tqdm
```

החבילה datasets מכילה קורפוסים לא-מתויגים רבים, בדיוק מהסוג שאנחנו צריכים לטובת אימון טוקניזר תת-מילי. נשתמש ב-[OSCAR](#), פרויקט נקיון ותחזוקה של מידע ממאגר הגריפה הגדול commoncrawl התומך ב-166 שפות, וביניהן עברית.

נבקש ממנו את מחיצת האימון (היחידה שקיימת) לעברית ונספק את הפרמטר streaming כדי שנוכל לדגום חלק קטן ממנה בלי להוריד הכל לזכרון של קולאב (המון זמן). 200,000 מסמכים יהיו די והותר.

```
dataset = datasets.load_dataset(
    'oscar',
    'unshuffled_deduplicated_he',
    split='train',
    streaming=True,
    trust_remote_code=True)

sampled_ds = []
for i, doc in tqdm(enumerate(dataset)):
    if i > 200_000:
        break
    sampled_ds.append(doc)
```

```
git rm .gitignore
```

Start coding or [generate](#) with AI.

היעלה על הדעת שנוריד מידע ולא נתבונן בו? הבה נבחן את המסמכים האחרונים שהורדנו.

sampled_ds[-10:]

```
[[{'id': 199991,
  'text': 'ולכבוד הסרט גוגל בנו משחק קצרצר של מלחמת הכוכבים הפועל "Star Wars The Force Awakens" בלילה האחרון החלו הקרנות הבכורה בארץ ובעולם של הסרט החדש.
Stormעליכם יהיה להכנס לקישור עם הקוד המיוחד שיוצג לכם באתר במחשב מהפלאפון ולאחר ביצוע כיוול קצרצר ופשוט תוכלו להדוף יריות ולבסוף גם להלהם בn. בתוך דפדפן הכרום
תגיות: גוגל, דפדפן, כרום, n. לצערי המשקה מאוד מאוד קצר ומכיוון שמדובר במשחק דפדפן באיכות לא רעה ושליטה שניעשת ממכשיר אחר דרך הרשת יש מדי פעם קצת האטהn.
Trooper. Chrome, Lightsaber, Star Wars, Storm Trooper, The Force Awakens',
  {'id': 199992,
  'text': 'התרגלנו לחשוב שיש לנו סוג של בעלות על האדם האהוב עלינו. לפעמים מספיק שנראה את הבן זוג שלנו מסתכל על אישה אחרת, והקנאה מתעוררת. האמירה הבאה אולי':
  {'id': 199993,
  'text': 'כולנו מנהלים את עצמינו במספר מעגלי חיים: קריירה, משפחה, חברה, תחומי עניין אישיים. לא פעם אנו מוצאים את עצמינו "נקרעים" בין ה"עולמות" הנ"ל ולא פעם':
  {'id': 199994,
  'text': 'הנוסע המוסלמי אבן בטוטא (1304-1377) ביקר בטורקיהnדף הבית << הרצאות << טורקיה << סקר ארכיאולוגיnהסטוריה של אתרים מרכזיים יווניים ורומיים במחוזות טורקיה':
  {'id': 199995,
  'text': 'אם זה לגובה או אם זה למרחב, בכל סוג של פרויקט מתקן החשמל דורש שיטות תכנון מתקדמות אשר התפתחו עם השנים מליבה יציבה ומוכחת של ידע, ניסיון ועבודת צוות.
  {'id': 199996,
  'text': 'אחד השיקולים החשובים ביותר בבחירת מקרר הוא בחירת הנפח המתאים. מקרר גדול מידי יגרם לבובזו אנרגיה ומקרר קטן מידי יהיה מוגבל ביכולת האחסון, 500-600 ליטרים':
  {'id': 199997,
  'text': 'סוויד - זמש הוא החומרים PU חומר עליוןnאביב / סתיו נעלים ללא שרוכים לבן / שחור 6534228 2018 - $29.99 PU / בגדי ריקוד גברים נעלי נוחות סוויד':
  {'id': 199998,
  'text': 'כדי לסייע לכם להתאים את השיש לבחירות השונות שצריך לעשות במסגרת תהליך של עיצוב מטבחים, נועד מאמר זה. השתמשו בו בתבונה והשיגו ממנו את המרב. אתם בהחלט':
  {'id': 199999,
  'text': 'דעו כי בכל הנוגע לבחירת שישn, יכולים לקחת את מה שמועיל לכם ולהתעלם ממה שלא נראה לכם רלוונטי, אך דעו כי הבחירה הנכונה היום, היא שתאפשר לכם ליהנות מהשיש מחר ובכלל
  {'id': 200000,
  'text': 'נשאלת השאלה החשובה מתי נכון לבחור את השיש והאם לתקופה בה מתבצעת הבחירה יש משמעות מסוימת, התשובה על כךn. השימוש במשטה, בטעם האישי שלכם מבחינה עיצובית ועוד
```

לפי הסרט המוזכר במסמך 199,991, עושה רושם שהדאטאסט, או לפחות החלק הזה שלו, הוא מאמצע העשור הקודם.

שמירת דאטאסט מסודר למטרות אימון טוקנייזר

התא הבא מכין את הדאטא עבור מנגנון האימון שכיר בחלק הבא.

CORPUS_DIR = './oscar_heb'

```
if not os.path.exists(CORPUS_DIR):
    os.mkdir(CORPUS_DIR)
```

```
text_data = []
file_count = 0
```

```
for sample in tqdm(sampled_ds):
```

```
# remove newline characters from each sample as we need to use exclusively as seperators
sample = sample['text'].replace('\n', '\s')
```


מי שמעוניינת מוזמנת לשמור את הקבצים בנפרד מהמחברת ולטעון לפי הצורך בשלבי פתרון התרגיל כדי לחסוך את זמן האימון עם כל ששן. עם זאת, ההגשה הסופית חייבת להיות תוצר של הרצה אחת נקייה מההתחלה ועד הסוף.

הבה נסתכל על כמה דוגמאות:

```
from transformers import BertTokenizer
```

```
MODEL_DIR = './wp-heb'
tokzr = BertTokenizer.from_pretrained(f'{MODEL_DIR}/wp-heb-vocab.txt')
```

```
print(f'There are {tokzr.vocab_size} tokens in our tokenizer\'s vocabulary.\n===')
print(tokzr.tokenize('hello world'))
print(tokzr.tokenize('שלום עולם'))
print(tokzr.tokenize('אם אין לי מי לי'))
```

```
c:\Users\Benzu\anaconda3\Lib\site-packages\transformers\tokenization_utils_base.py:1925: FutureWarning: Calling BertTokenizer.from_pretrained
warnings.warn(
There are 3000 tokens in our tokenizer's vocabulary.
===
['h', '##el', '##l', '##o', 'w', '##or', '##l', '##d']
['שלו', '##ם', 'עו', '##ם']
['אם', 'אין', 'אני', 'לי', 'מי', 'לי']
```

כמו שאנחנו רואים, הטוקנייזר שלנו בהחלט יודע לעבד עברית יותר טוב מאשר אנגלית, בהתאם לדאטא שקיבל לאימון.

סעיף 1 (0.5 נק): הצינו הסבר מדוע בכל-זאת "שלום" מפריד את האות הסופית אך "עולם" מפריד שתי אותיות אחרונות. ניתן להוסיף תא קוד להנמקה.

הסבר אפשרי הינו ש'שלו' הינו נפוץ יחסית ולכן קיבל טוקן משל עצמו, מאחר וזהו אלגוריתם גרידי הוא תחילית ארוכה יותר מ'של' ואילו 'עול' ככל הנראה לא נפוץ מספיק בשביל לקבל טוקן משל עצמו ולכן 'עו' קיבל את התחילית הארוכה ביותר (למטה ניתן לראות שעולים ועושים שניהם מתפרקים לטוקן 'עו')

```
print(tokzr.tokenize('שלום עולם'))

print(tokzr.tokenize('את לא כמו כולם'))
print(tokzr.tokenize('אין מקום בעולם בשבילם'))
print(tokzr.tokenize('סתם הולם'))

print(tokzr.tokenize('עולים לגובה עושים הרבה'))
print(tokzr.tokenize('שלחתי שליחים לשלום שלומית'))
```

```
['שלו', '##ם', 'עו', '##ם']
['את', 'לא', 'כמו', 'כול', '##ם']
['אין', 'מקום', 'בעולם', 'בש', '##ביל', '##ם']
['ס', '##תם', 'הל', '##ום']
['עו', '##לים', 'לג', '##ובה', 'עו', '##שים', 'הרבה']
['של', '##הת', '##י', 'שלי', '##חים', 'לש', '##לו', '##ם', 'שלו', '##מית']
```

כפי שאנו זוכרים מהשיעור, לרוב מאמנים טוקנייזר על אוצר מילים בסדר גודל של 30-50 אלף, ושלנו קטן פי עשרה. התוצאה היא הרבה פחות מילים וחלקי מילים שימושיים שזכו לקבל תמנית משלהן.

סעיף 2 (0.5 נק): מצאו מילה (רצף אותיות עבריות ללא רווחים) תקינה בעברית שמתפצלת ליותר מ-6 תמניות.

```
### 3 ###
MY_WORD = 'האינצקלופדיה'
### # ###

toks = tokzr.tokenize(MY_WORD)
print(len(toks), toks)

['האי', '##נ', '##צ', '##קל', '##יפ', '##די', '##ה']
```

כפי שהזכרנו בכיתה, אופן הפענוח (decoding, inference) של אלגוריתם וורדפיס הינו חמדני (greedy): בהינתן רצף תווים, וורדפיס יחפש את התמנית הארוכה ביותר שמתאימה לתחילתו, וימשיך מהנקודה בה נקטע.

סעיף 3 (1 נק): ממשו את אלגוריתם הפענוח באופן נכון אך לא בהכרח יעיל, באמצעות אובייקט הקורא את אוצר המילים ופועל חמדנית. הקוד בתא השני יבדוק את נכונותו.

```
from typing import Callable, List

class GreedyTokenizerFromFile(Callable):
```

```
def __init__(self, vocab_file: str) -> None:
    self.vocab = set()

    ### 5.1 (0.25 points) ###
    self.vocab |= set(open(vocab_file, 'r', encoding='utf-8').read().splitlines())

    print(f'initialized with {len(self.vocab)} pieces')

def __call__(self, s: str) -> List:

    ### 5.2 (0.75 points, conditioned on test passing) ###
    tokens = []
    word = s
    while word != '###':
        for i in range(len(word), 0, -1):
            if word[:i] in self.vocab:
                tokens.append(word[:i])
                word = '###' + word[i:]
                break
        elif i == 1:
            tokens.append(['UNK'])
            return tokens

    return tokens

### you may define any additional functions below ###

#####
### Do not modify this cell when submitting ###
#####

from tokenizers.pre_tokenizers import BertPreTokenizer

greedy_tok = GreedyTokenizerFromFile(f'{MODEL_DIR}/wp-heb-vocab.txt')

# Bert's PreTokenizer handles punctuation. It's called implicitly by tokzr
pretok = BertPreTokenizer()

# our "test set" is a random document from the dataset
for w, char_span in tqdm(pretok.pre_tokenize_str(sampled_ds[2023]['text'])):

    # ground truth - the automatically-learned tokenization
    trained_tok = tokzr.tokenize(w)

    # preparation - BertTokenizer actually has a preprocessing submodule called
    # "basic_tokenizer" that lowercases English and such
    grd_tok = greedy_tok(tokzr.basic_tokenizer.tokenize(w)[0])

    # Test - these asserts should not fail #
    assert trained_tok == grd_tok, f'{w}, {trained_tok}, {grd_tok}'
```

```
→ initialized with 3000 pieces
a%| | a/25 [aa-aa<? >it+<]
```

משחק השבת

נניח שאנחנו מתכננים משימה שנוגעת לעיבוד טקסט ממסמכים הנוגעים לספורט. בואו נראה עד כמה המודל שלנו מותאם למילים מתוך התחום (domain) הספורטיבי:

```
' '.join(tokzr.tokenize('מכבי הפועל כדורגל כדורסל כדורעף טניס'))
```

```
→ 'מכבי הפועל כדורגל כדורסל כדורעף טניס'
      מכבי הפועל כדורגל כדורסל כדורעף טניס
```

נראה שהמצב די רע. ננסה לאמן מודל טוקניזציה שיטפל במילים האלה יותר טוב רק באמצעות שינוי קורפוס האימון.

סעיף 4 (2 נק', אין ניקוד חלקי): צרו דאטאסט חדש מתוך הקורפוס שכבר בדינו (אל תייבאו קורפוס חדש, אל תפצלו מסמכים, ואל תכתבו יותר מ-200,000 מסמכים לדיסק) ואמנו עליו מודל WordPiece עם אותם היפר-פרמטרים כמקודם, שמפצל לכל היותר מילה אחת מהמילים לעיל ליותר משתי תמויות, ולא מפצל בכלל לפחות שתיים מהמילים. הראו את הפלט. תעדו היטב באמצעות הערות את כל השינויים שערכתם ביחס לקוד המקורי.

צוותים שיפנו בשעות הקבלה שלפני תחילת השיעור ב-7/1 עם פתרון חלקי יוכלו לקבל רמז. לאחר מועד זה לא תינתן עזרה על הסעיף.

```
import re

def clean_text(text: str) -> str:
    """
    Cleans the input text by removing characters that are not Hebrew letters, digits, spaces, or specified punctuation/symbols.
    Args:
        text (str): The input text to be cleaned.
    Returns:
        str: The cleaned text with only allowed characters and no extra spaces.
    """

    hebrew_digits_punct_regex = re.compile(r"[^\d-א-z\s.,!?:'\"`(){}[\]\|-+*/@#%&*;<>~|]")
    cleaned_text = hebrew_digits_punct_regex.sub("", text)
    cleaned_text = re.sub(r'\s+', ' ', cleaned_text).strip()
    return cleaned_text

text = "45 ()*&%$#@ (: האם זה עובד? ,123 והו מבחן abc <>"
cleaned_text = clean_text(text)
print(cleaned_text)
```

➡ 45 <> שלום! זהו מבחן 123, האם זה עובד? (: @%\$^&()*

```
def download_corpus(keywords: List[str], n_docs: int) -> List[dict]:
    """
    Downloads a corpus of documents containing specified words.
    This is the same as before, just filtering the documents by specified keywords.
    Args:
        words (List[str]): A list of words to search for in the documents.
        n_docs (int): The number of documents to download.
    Returns:
        List[dict]: A list of dictionaries, each representing a document that contains any of the specified words.
    """
    i = 0
    sampled_sports_ds = []

    with tqdm(total=n_docs, desc="Finding sports documents") as pbar:
        for doc in dataset:
            if any([word in doc['text'] for word in keywords]):
                sampled_sports_ds.append(doc)
                i += 1
                pbar.update(1)
                if i >= n_docs:
                    break
    return sampled_sports_ds
```

```
import shutil
```

```
def save_corpus(corpus: List[dict], directory: str, clean=False) -> None:
    """
    Save a cleaned corpus to a specified directory in chunks of 5000 samples.

    Args:
        corpus (List[dict]): A list of dictionaries, each containing a 'text' key with the text sample.
        directory (str): The directory where the cleaned text files will be saved.

    Returns:
        None
    """

    if os.path.exists(directory):
        shutil.rmtree(directory)
    os.mkdir(directory)

    text_data = []
    file_count = 0

    for sample in tqdm(corpus, desc="Saving corpus"):

        # remove newline characters from each sample as we need to use exclusively as separators
        sample = sample['text'].replace('\n', '\s')

        text_data.append(sample)
        if len(text_data) == 5_000:

            # once we hit the 5K mark, save to file
```

```
with open(f'{directory}/text_{file_count}.txt', 'w', encoding='utf-8') as fp:
    text = clean_text('\n'.join(text_data)) if clean else '\n'.join(text_data)
    fp.write(text)
```

```
text_data = []
file_count += 1
```

```
# after saving in 5K chunks, we may have leftover samples, we save those now too
with open(f'{directory}/text_{file_count}.txt', 'w', encoding='utf-8') as fp:
    text = clean_text('\n'.join(text_data)) if clean else '\n'.join(text_data)
    fp.write(text)
```

```
>>:24: SyntaxWarning: invalid escape sequence '\s'
>>:24: SyntaxWarning: invalid escape sequence '\s'
C:\Users\Benzu\AppData\Local\Temp\ipykernel_35672\3057535395.py:24: SyntaxWarning: invalid escape sequence '\s'
sample = sample['text'].replace('\n', '\s')
```

```
SPORTS_WORDS = 'מכבי הפועל כדורגל כדורסל כדורעף טניס'.split()
SPORTS_CORPUS_DIR = './sports_adjacent-clean'
SPORTS_MODEL_DIR = './sp-heb'
MODEL_NAME = 'sports_adjacent-clean'
N_DOCS = 50_000
```

```
### 4 ###
```

```
# download the corpus with the sports keywords
sampled_sports_ds = download_corpus(SPORTS_WORDS, N_DOCS)
```

```
# clean and save the corpus
save_corpus(sampled_sports_ds, SPORTS_CORPUS_DIR, clean=True)
```

```
# All of this is the same as before, but with the new corpus
sports_tokzr = tokzr_sports_trainer = BertWordPieceTokenizer(clean_text=True)
sports_paths = sorted([str(x) for x in Path(SPORTS_CORPUS_DIR).glob('**/*.txt')])
tokzr_sports_trainer.train(files=sports_paths, vocab_size=3_000, min_frequency=2,
                           limit_alphabet=1_000, wordpieces_prefix='##',
                           special_tokens=[
                               '[PAD]', '[UNK]', '[CLS]', '[SEP]', '[MASK]'])
if not os.path.exists(MODEL_DIR):
    os.mkdir(MODEL_DIR)
```

```
tokzr_sports_trainer.save_model(MODEL_DIR, f'{MODEL_NAME}')
```

```
Finding sports documents: 0%|          | 0/50000 [00:00<?, ?it/s]
Saving corpus: 0%|          | 0/50000 [00:00<?, ?it/s]
['./wp-heb\sports_adjacent-clean-vocab.txt']
```

```
sports_tokzr = BertTokenizer.from_pretrained(f'{MODEL_DIR}/{MODEL_NAME}-vocab.txt')
```

```
### # ###
```

```
' '.join(sports_tokzr.tokenize('מכבי הפועל כדורגל כדורסל כדורעף טניס'))
```

```
c:\Users\Benzu\anaconda3\Lib\site-packages\transformers\tokenization_utils_base.py:1925: FutureWarning: Calling BertTokenizer.from_pretr
warnings.warn(
'מכבי הפועל כדורגל כדורסל כדורעף טניס'
### # ###
```

