THE;INSTITUTE
Empowering Israel's Future Through AI

AI Developers

Practice

Tokenization

תהליך חלוקה של משפט לtokens, שלב לפני הtransformer.

basic_tokenizer.py

השתמשו בטוקנייזר bert-base-uncased, ופרקו משפט לtokens.
לאחר מכן, תרכיבו חזרה את הtokens להיות משפט.
השתמשו בספריית transformers

compare_tokenizer.py

השתמשו בטוקנייזרים שונים על משפט זהה.
מה ההבדלים ביניהם?

bert-base-uncased, roberta-base, xlm-roberta-base

Model	Tokenizer Type	Approach
bert-base-uncased	WordPiece	Greedy, deterministic merging of the most frequent subwords, using **likelihood** based on training corpus
roberta-base	BPE (Byte-Pair Encoding)	Starts with characters, repeatedly merges the **most frequent adjacent character pairs**
xlm-roberta-base	SentencePiece	Trains on raw text using a **Unigram Language Model**, chooses token splits probabilistically for highest likelihood segmentation, with no pre-tokenization

TF-IDF

מדידת חשיבות מילים בתוך מסמך.
TF = Term Frequency
IDF = Inverse Document Frequency

Term Frequency (TF)

$$TF(t,d) = \frac{Apperance\ count\ of\ word\ t\ on\ document\ d}{Amount\ of\ words\ in\ d}$$

למשל, במסמך d של 100 מילים, אם המילה "פלאפל" מופיעה 5 פעמים:

$$TF(\text{"פלאפל"}, d) = \frac{5}{100} = 0.05$$

Inverse Document Frequency (IDF)

$$IDF(t, D) = \log\left(\frac{Total\ number\ of\ documents\ in\ corpus\ D}{Amount\ of\ documents\ containing\ the\ word\ t}\right)$$

למשל, בקורפוס D של 300 כתבות אוכל, אם המילה
"פלאפל" מופיעה ב3 כתבות :

$$IDF(\text{"פלאפל"}, D) = \log\left(\frac{300}{3}\right) = \log(100) = 2$$

Inverse Document Frequency (IDF)

לעומת זאת, באותו קורפוס של 300 כתבות אוכל, המילה
"אוכל" מופיעה ב299 כתבות :

$$IDF(\text{"אוכל"}, D) = \log\left(\frac{300}{299}\right) = 0.00145$$

Inverse Document Frequency (IDF)

ככל שהIDF יותר קטן, זה אומר שהמילה נפוצה יותר בקורפוס. במקרה שלנו:

- הIDF של "פלאפל" נחשב יחסית גדול, משמע המילה נדירה יחסית בקורפוס.
- הIDF של "אוכל" קטן מאוד (שואף ל0), משמע המילה נפוצה בקורפוס.

$$IDF(\text{"פלאפל"}, D) = 2$$

$$IDF(\text{"אוכל"}, D) = 0.00145$$

*מה קורה אם המילה מופיעה בכל מסמך בקורפוס?

TF-IDF

את הTF-IDF נחשב על ידי מכפלה של הTF בIDF:

$$TF - IDF(t, d, D) = TF(t, d)\ x\ IDF(t, D)$$

נחשב את הTF-IDF עבור הדוגמאות שראינו (נניח שהמילה "אוכל" מופיעה גם היא 5 פעמים במאמר שבחרנו):

$$TF - IDF(\text{"פלאפל"}, d, D) = 2\ x\ 0.05 = 0.1$$

$$TF - IDF(\text{"אוכל"}, d, D) = 0.00145\ x\ 0.05 = 0.00007$$

tfidf_test.py

נשתמש בTfidfVectorizer מתוך sklearn.
תיצרו רשימת מחרוזות (כל מחרוזת תהיה מסמך, הרשימה היא הקורפוס שלנו).
תחשבו את הTF-IDF עבור כל מילה במסמכים, והציגו את התוצאות.

tfidf_search.py

ניקח רשימה של מחרוזות, ונבצע חיפוש של מחרוזת query. עלינו למצוא את המסמך הכי רלוונטי למחרוזת query.

נשתמש בtf-idf וcosine-similarity.

NextTokenPredict.py

נשתמש במודל GPT-2 על מנת לחזות
משפטים על סמך תחילתם.
נשתמש בספריית transformers

Prompt Engineering תרגול

1 – Zero-Shot, One-Shot, Few-Shot

Prompt Engineering תרגול

2 – Prompt Tuning

THE;INSTITUTE

תודה!