עמוד שער

מגיש: יוני קרמר

תעודת זהות: 215005737

בית ספר: עירוני ד' על שם אהרון קציר, תל אביב

מנחה: עידו גודיס

רכזת עבודות גמר: לימור שיאון

תוכן עניינים

[עמוד שער 1](#_Toc122779739)

[תוכן עניינים 2](#_Toc122779740)

[הקדמה אישית ותודות 4](#_Toc122779741)

[מבוא 4](#_Toc122779742)

[שאלת המחקר: 5](#_Toc122779743)

[סקירת בינה מלאכותית 4](#_Toc122779744)

[מושגי בסיס: 5](#_Toc122779745)

[למידת מכונה (או למידה חישובית) מונחית – Supervised Machine Learning: 5](#_Toc122779746)

[למידה בהנחיה עצמית – Self Supervised Learning: 5](#_Toc122779747)

[למידה בלתי מונחית – Unsupervised Learning: 5](#_Toc122779748)

[ווקטור הסתברות – Probability Vector: 5](#_Toc122779749)

[ווקטור לוג'יט – Logit Vector: 5](#_Toc122779750)

[טנזור - Tensor: 5](#_Toc122779751)

[טוקניזציה – Tokenization: 5](#_Toc122779752)

[**Dummy/One Hot Encoding:** 5](#_Toc122779753)

[Cross Entropy 5](#_Toc122779754)

[**שיכון טוקנים – Token Embedding:** 5](#_Toc122779755)

[**הפונקציה softmax:** 5](#_Toc122779756)

[**טמפרטורה - Temprature:** 5](#_Toc122779757)

[מודל שפה סיבתי – Cuasal Language Model: 5](#_Toc122779758)

[טרנספורמר עם דיקודר בלבד – Decoder Only Transformer: 5](#_Toc122779759)

[בלוק דיקודר Decoder Block: 5](#_Toc122779760)

[קידוד מיקומי – Positional Encoding: 5](#_Toc122779761)

[הכפלה במטריצת שיכון משוחלפת – Transposed Embedding: 5](#_Toc122779762)

[מסכת הסתקלות קדימה look ahead mask: 5](#_Toc122779763)

[צומת לב עצמית ממוסכת בעזרת מכפלה סקלרית: 5](#_Toc122779764)

[צומת לב רב-ראשית – Multi Head Attention: 5](#_Toc122779765)

[רשת מחוברת לגמרי – Fully Connected Feed Forward Network: 5](#_Toc122779766)

[שכבה דחוסה\מחוברת לגמרי – Dense/Fully Connected Layer: 5](#_Toc122779767)

[פונקציית ReLU: 5](#_Toc122779768)

[ריפוד - Padding: 5](#_Toc122779769)

[**אימון ראשוני למידול שפה סיבתי בעזרת יצירת שפה:** 5](#_Toc122779770)

[**Generative Pre-Training (GPT) for Causal Language Modeling** 5](#_Toc122779771)

[**למידה רב שלבית – Transfer Learning:** 5](#_Toc122779772)

[דגימה – Sampling/Decoding: 5](#_Toc122779773)

[דגימה לפי הסתברות מקסימלית – Argmax/Greedy Sampling: 5](#_Toc122779774)

[דגימה מתוך הסתברות – Pure Sampling: 5](#_Toc122779775)

[דגימה מתוך k הטוקנים שהסתברותם הכי גבוה – Top k Sampling: 5](#_Toc122779776)

[דגימה מתוך הטוקנים שסכום הסתברותם <= p – Top p Sampling: 5](#_Toc122779777)

[חיפוש עץ – Tree Sampling: 5](#_Toc122779778)

[חיפוש עץ עם אילוץ על סכום ההסתברות: 5](#_Toc122779779)

[דגימה עם עונשים - penalized sampling: 5](#_Toc122779780)

[מדד ברט – BERT Score: 5](#_Toc122779781)

[הנדסת פרומפט – Prompt Engeneering: 5](#_Toc122779782)

[סקירת פייתון 4](#_Toc122779783)

[רשימת מבני נתונים בפייתון ומבני הנתונים המקבילים בשפות אחרות: 5](#_Toc122779784)

[רמזי סוג – Type Hints: 5](#_Toc122779785)

[רב צורתיות (פולימורפיזם): 5](#_Toc122779786)

[פעולות קסם – Magic/Dunder Methods: 5](#_Toc122779787)

[קשטנים - :Decorators 5](#_Toc122779788)

[מחלקת בסיס אבסטרקטית: 5](#_Toc122779789)

[פונקציה אבסרקטית: 5](#_Toc122779790)

[פעולה סטטית: 5](#_Toc122779791)

[מחלקת נתונים – Data Class: 5](#_Toc122779792)

[המחלקה האבסרקטית Callable: 5](#_Toc122779793)

[Enum: 5](#_Toc122779794)

[ערימת מקסימום\מינימום – Maximum/Minimum Heap: 5](#_Toc122779795)

[פיתוח מודלי למידה עמוקה בעזרת tensorflow: 5](#_Toc122779796)

[המחלקה Module: 5](#_Toc122779797)

[המחלקה Layer: 5](#_Toc122779798)

[שכבות בנויות מראש: 5](#_Toc122779799)

[המחלקה Model: 5](#_Toc122779800)

[פיתוח התכונה 4](#_Toc122779801)

[האלגוריתם ליצירת טקסט בעזרת קבוצות של טוקנים: 5](#_Toc122779802)

[המחלקה :TextGenerator 5](#_Toc122779803)

[המחלקה TreeGenerator: 5](#_Toc122779804)

[המחלקה SampleGenerator: 5](#_Toc122779805)

[אפליקציית הווב: 5](#_Toc122779806)

[המחלקה Blueprint: 5](#_Toc122779807)

[המשתנה הגלובלי :g 5](#_Toc122779808)

[**הקובץ \_\_init\_\_.py:** 5](#_Toc122779809)

[הקובץ database.py: 5](#_Toc122779810)

[הקובץ schema.sql: 5](#_Toc122779811)

[הקובץ auth.py: 5](#_Toc122779812)

[הקובץ model.py: 5](#_Toc122779813)

[מימוש הארכיטקטורה לטרנספורמר עם דיקודר בלבד: 5](#_Toc122779814)

[המחלקה Transformer: 5](#_Toc122779815)

[שאר המחלקות: 5](#_Toc122779816)

[הפונקציה positional\_encoding: 5](#_Toc122779817)

[הפונקציה create\_masks: 5](#_Toc122779818)

[הדמו: 5](#_Toc122779819)

[הצגת התוצרים 4](#_Toc122779820)

[בדיקת איכות הטקסטים שנוצרים על ידי האלגוריתם: 5](#_Toc122779821)

[תוצאות – חלק ראשון 5](#_Toc122779822)

[ביבליוגרפיה 4](#_Toc122779823)

[נספחים 5](#_Toc122779824)

[חלקים נבחרים מהקוד: 5](#_Toc122779825)

[דיאגרמת בסיס הנתונים של אפליקציית הווב: 5](#_Toc122779826)

[קישורים: 6](#_Toc122779827)

[דוקומנטציות: 6](#_Toc122779828)

[קורסים של אוניברסיטת סטנפורד: 6](#_Toc122779829)

[מדריכים: 6](#_Toc122779830)

הקדמה אישית ותודות

מבוא

שאלת המחקר:

האם ניתן ליצור טקסט באורך n טוקנים בעזרת פחות מ n שימושים מודל שפה סיבתי (causal language model)? אם כן, כיצד? ואיך שימוש באלגוריתם זה משפיע על הטקסט שנוצר? (בהשוואה לאלגוריתם שיוצר n טוקנים על ידי n קריאות למודל).

סקירת בינה מלאכותית

מושגי בסיס:

למידת מכונה (או למידה חישובית) מונחית – Supervised Machine Learning:

שם כולל לאלגוריתמים שמקבלים נתונים בצורת צמדי קלט ופלט רצוי ומטרתם להגדיל או להקטין ערך של פונקציה מסוימת (פונקציית המטרה) המחשבת את הדמיון או השוני בין פלט האלגוריתם לפלט הרצוי.

לדוגמה: מודל שמסווג תמונות למספר קבוע של מחלקות ידועות.

למידה בהנחיה עצמית – Self Supervised Learning:

למידת מכונה בה האלגוריתם מקבל נתונים שאינם מחלוקים לצמדי קלט ופלט וחלקוה זו נעשית על ידי האלגוריתם עצמו.

לדוגמה – מודל שמקבל טקסט וחוזה את המילה הבאה.

למידה בלתי מונחית – Unsupervised Learning:

למידת מכונה בה אין פלט רצוי ופונקציית המטרה מחשבת מטרה אחרת.

לדוגמה – רכב אוטונומי שפונקציית המטרה שלו מחשבת את הסיכוי לתאונה.

ווקטור הסתברות – Probability Vector:

במדעי הנתונים –הוא ווקטור בו P\_i מייצג את ההסתברות של מחלקה i. כל ההסתברויות בין אפס לאחד וסכום הווקטור 1.

ווקטור לוג'יט – Logit Vector:

בלמידה עמוקה – ווקטור לוג'יט הוא ווקטור בו מוצגות הסתברויות בטווח מינוס אינסוף עד אינסוף. פונקציית Softmax משמשת (בין היתר) להפוך ווקטור לוג'יט לווקטור הסתברות.

טנזור - Tensor:

מערך רב מימדי של מספרים. ניתן לחשוב על טנזור כעל מפה בין מספרים שלמים ואי שליליים לערכים מכל צורה.

לדוגמה:

נניח ויש לנו אתר בו כל פעם שמשתמש נכנס לאתר נוצרת טבלה (מטריצה) בתוך טנזור שמתארת את פעילותו באתר.

יתאר את התא בשורה ועמודה בטבלה שמתארת את הכניסה מספר של המשתמש מספר .

יתאראת העמודה ה בטבלה שמתארת את הכניסה מספר של המשתמש מספר

וככה הלאה.

הדרגה\רמה\מעלה של טנזור מתארת את מספר הפרמטרים המקסימלי של מספרים שאפשר להעביר אליו.

טנזור ברמה אפס הוא מספר (לדוגמה מינוס שבע עשרה) וזאת מכיוון שאי אפשר לגשת לאיבר מספר שלוש של מינוס שבע עשרה.

טנזור ברמה אחת הוא ווקטור או מערך של מספרים (לדוגמה ) וזאת מכיוון שאפשר לגשת לאיבר מספר שתיים במערך אך אי אפשר לגשת לאיבר כלשהו של המספר חמש עשרה.

נשים לב כי טנזור ברמה אחת הוא מערך של טנזורים ברמה אפס.

טנזור ברמה שתיים הוא טבלה או מטריצה וזאת מכיוון שטבלה היא מערך של עמודות (או שורות) שהן בעעמן טנזורים מרמה אחת.

באופן כללי, נגדיר טנזור ברמה אפס כמספר וטנזור ברמה ( מספר טבעי) כמערך של טנזורים ברמה באותה צורה ואותו סוג נתונים.

טוקניזציה – Tokenization:

בניגוד לתוכנות קלאסיות (שלא משתמשות בלמידת מכונה) מודלי שפה אינם מיצגים טקסט כרצף אותיות (מחרוזת) אלא כרצף של מילים, חלקי מילים או צירוף אותיות בעל משמעות (למשל הסיומת ים לציון רבים בעברית או הסיומת ing באנגלית).

כשיוצרים טוקנייזר וקובעים את גודל המילון, הוא מוצא את רצפי האותיות הכי נפוצים בסט האימון ונותן לכל אחד מהם מזהה (טוקן) בצורת מספר שלם ואי שלילי.

לטוקנייזר שתי פונקציות מרכזיות:

Encode: הטוקנייזר מקבל מחרוזת ומחזיר רצף של מזהיי טוקן לפי הסדר בהם הם מופיעים בטקסט כרשימה או כטנזור מסוג int.

Decode: הטוקנייזר מקבל רצף של מזהיי טוקן ומחזיר ומתרגם אותם למחרוזת.

כל מודל שקולט ו\או מייצר שפה מאומן בהינתן טוקנייזר – הטוקנייזר מוגדר לפני תחילת אימון המודל ולא משתנה אף פעם. שימוש במודל שפה בעזרת טוקנייזר לא מתאים או שינוי של הטוקנייזר גורם לתוצאות חסרות משמעות.

**Dummy/One Hot Encoding:**

(Suits)

*יצוג one hot ממיר אינדקס של מחלקה (במקרה שלנו טוקן) שהוא מספר שלם ואי שלילי לווקטור\מערך\רשימה בינארית שאורכה כמספר המחלקות בנתונים (במקרה שלנו מספר הטוקנים שהטוקנייזר שומר במילון).*

*יצוג one hot הוא בעצם ווקטור של הסתברות של תוצאה ידועה מראש ולכן אנחנו רוצים שההסתברות שהמודל יחזה תיהיה כמה שיותר קרובה ליצוג one hot .*

|  |
| --- |
| *def token\_to\_one\_hot(token\_id: int, num\_tokens: int) -> List[int]:  """Returns a one-hot list for the class class\_id with*  *num\_classes classes"""  ans: List[int] = [0] \* num\_tokens  ans[token\_id] = 1  return ans   def sequence\_to\_one\_hot(sequence: List[int],  num\_tokens: int) \  -> List[List[int]]:  """Creats a one-hot matrix  for the given sequence of tokens"""  return [token\_to\_one\_hot(token\_id, num\_tokens)  for token\_id in sequence]* |

לדוגמה:

אם המילון של הטוקנייזר הוא {אני: 0, אוהב: 1, גלידה: 2} אז המחזרות "אני אוהב גלידה" תומר לרצף הטוקנים [2, 1, 0] ואז לייצוג one hot:

Cross Entropy

פונקציה המודדת את המרחק בין 2 ווקטורי הסתברות .

אם הוא ווקטור onehot המיצג את המחלקה עם אינדקס , ניתן להשתמש בפונקציה השקולה:

שהסיבוכיות שלה היא .

פסודו קוד:

float Sparse Categorical Cross Entropy(p: Tensor[n], q: OneHotVector[n])

c: int = q.getindex()

return -math.log(p[c])

**שיכון טוקנים – Token Embedding:**

(Mikolov et al.)

שיכון היא שכבה המקבלת רצף טוקנים ביצוג one hot וממירה כל יצוג one hot בווקטור בעל משעות סמנטית שאינו תלוי בטוקנים אחרים.

השכבה לומדת מטריצה בגודל מספר הטוקנים המוכרים למודל x שאפשר לדמיין אותה בתור טבלה בה כל שורה היא מילה שהמודל מכיר וכל עמודה היא תכונה סמנטית שיכולה להיות למילה או לצירוף אותיות בעל משמעות.

פעולת השיכון היא מכפלה של מטריצת הרצפים ביצוג one hot במטריצת השיכון.

**הפונקציה softmax:**

*(Bridle)*

*softmax() מוגדרת לפי הנוסחה:*

*תפקיד הפונקציה לקחת ווקטור עם ערכים בין מינוס אינסוף לאינסוף ולהפוך אותו לווקטור שערכיו בין אפס לאחד וסכומו אחד.*

*אחד השימושים שלה הוא חישוב ווקטור הסתברות מתוך ווקטור לוג'יט.*

*הווקטור שהפונקציה מחזירה הוא באותו גדול של הווקטור שהפונקציה מקבלת.*

*הפונקציה מחזירה ווקטור שסכומו אחד וכל איבריו בין אפס לאחד .*

*שימו לב גם ש .*

*פעולת softmax על מטריצה בציר מסוים היא פעולת softmax על כל ווקטור בציר.*

*כלומר: פעולת softmax בציר 1 היא פעולת softmax לכל שורה במטריצה.*

**טמפרטורה - Temprature:**

*(Ackley et al. 5)*

*טמפרטורה היא מספר חיובי שנבחר בעת יצירת טקסט ומטרתו להשפיע על התפלגות ווקטור ההסתברות שהמודל מיצר בדגימה שאיננה לפי הסתברות מקסימלית.*

*כל איבר במטריצת הלוג'יט מחולק בטמפרטורה לפני ככה ש:*

*אינטצואיציה:*

*הקטנת הטמפרטורה גורמת לאיזון ההסתברויות של הטוקנים ככה שלכל הטוקנים תיהיה הסתברות דומה יותר והגדלת הטמפרטורה גורמת לחוסר איזון בהסתברויות ככה שלטוקנים יהיו הסתברויות שונות יותר.*

*דוגמה: מציאת ווקטור ההסתברות של ווקטור הלוג'יט [1, 1-] עם טמפרטורה של אחד, שתיים וחצי.*

טמפרטורה איינה שימושית במהלך אימון המודל מיכוון שאם נשנה את הטמפרטורה, המודל ילמד ליצר ווקטור לוג'יט בו כל ערך מוכפל באחד חלקי הטמפרטורה.

מודל שפה סיבתי – Cuasal Language Model:

(Liu et al.)

מודל למידה עמוקה המקבל רצף של טוקנים  ומחזירה ומטריצת הסתברות בה הוא ווקטור ההסתברות שמייצג את החיזוי של המודל לטוקן במקום i+1 כאשר תלוי בטוקנים בלבד. בגלל שהלוג'יט של כל טוקן תלוי רק בטוקנים שבאו לפניו, המודל חוזה את הטוקן הבא ברצף.

המודל חוזה בין היתר את הטוקן שמגיע לאחר סוף הרצף שהוא מקבל וככה אפשר להשתמש במודל שפה סיבתי על מנת לחזות את הטוקן הבא במשפט.

היתרון של מודלי שפה סיבתיים הוא שבהינתן רצף טוקנים שאיננו מחולק לקלט ופלט ניתן לאמן את המודל לחזות את המילה הבאה בטקסט על ידי שימוש אחד בלבד במודל:

המודל מקבל מטריצה המייצגת רצף של טוקנים בצורת one-hot.

נוציא מהקלט את הטוקן במקום 0 כי המודל חוזה את הטוקנים החל מהמקום ה 0 ונוסיף טוקן מיוחד לסוף ההקלט שמעיד על סוף הרצף. עכשיו קיבלנו מטריצה חדשה בה:

נשים לב ש מייצג את הטוקן שנמצא במקום i במטריצה M ולכן נאמן את המודל לצמצם את המרחק בין M ו P .

טרנספורמר עם דיקודר בלבד – Decoder Only Transformer:

(Liu et al.)

ארכיטכטורה למימוש מודלי שפה סביתיים בה המודל מקבל רצף של מספרים שלמים (מזהיי טוקן), הופך אותם לשיכונים ומוסיף להם קידוד מיקומי ואת התוצאה מעביר לדיקודר.

הדיקודר מורכב מכמה בלוקים הנקראים בלוקי דיקודר כאשר הבלוק הראשון מקבל את הפלט לדיקודר ושאר הבלוקים מקבלים את הפלט של הדיקודר הקודם להם. הפלט של בלוק הדיקודר האחרון הוא הפלט של הדיקודר והוא מוכפל במטריצת אמבידנג משוחלפת ליצירת מטריצת לוג'יט.

על מטריצת הלוג'יט מופעלת פעולת שיוצרת את מטריצת ההסתברות שבה משתמשים על מנת ליצור רצף של טוקנים.

המודל מקבל מטריצה של ייצוג רצף הטוקנים בצורת one hot ומחזיר חיזוי בצורת לוג'יט של איזה טוקן יהיה בכל מקום ביחס בהתבסס על הטוקנים שבאים לפניו בטקסט.

בעבודה אשתמש במושג מטריצת הסתברות לתאר מטריצה בה היא ההסתברות של הטוקן ה ברצף להיות ה הטוקן שהמזהה שלו הוא i בהתבסס על הטוקנים הקודמים (אפס עד t לא כולל t). ומטריצת לוג'יט לתאר מטריצה הבנויה מווקטורי לוג'יט באותה הצורה.

בלוק דיקודר Decoder Block:

(Vaswani et al., pt.3.1)

בלוקי דיקודר הם שכבות עם תתי השכבות הבאות:

1. צומת לב עצמית רב ראשית עם מסכת הסתכלות קדימה
2. חיבור ונורמליזציה
3. רשת מחוברת לגמרי
4. חיבור ונורמליזציה

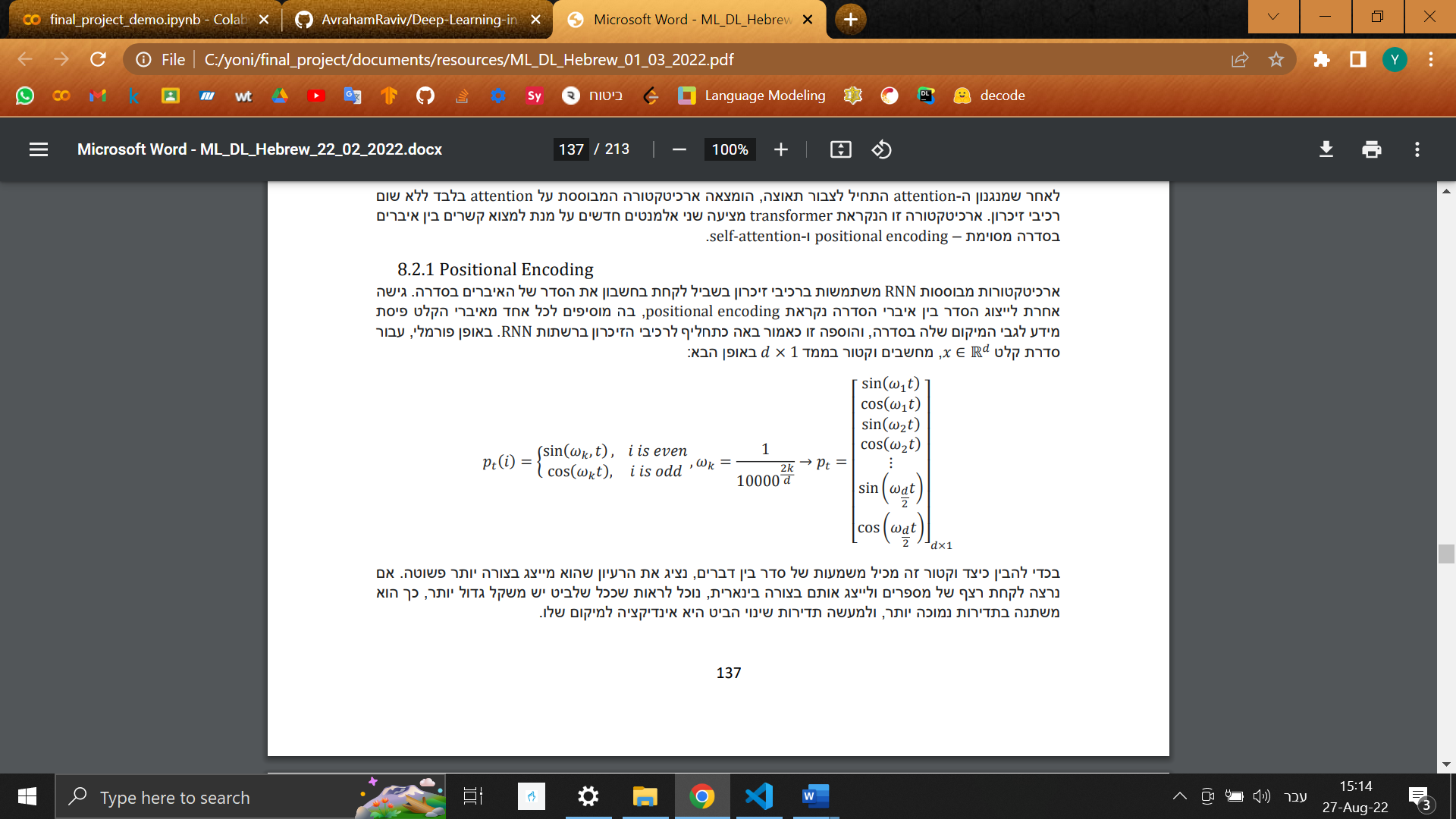
קידוד\שיכון מיקומי – Positional Encoding\Embedding:

(Vaswani et al., pt.3.5)

מכיוון שהטרנספורמר אינו מתייחס באופן שונה לווקטורים במיקומים שונים בתוך רצף, יש צורך בהוספת מידע לכל טוקן בנוגע למיקומו במשפט.

שיטה בה מוסיפים לכל אחד מאיברי הקלט פיסת מידע (במקרה שלנו טוקן) לגבי המיקום שלה ברצף באופן פורמלי, עבור סדרת קלט *באורך ניצור מטריצה בגודל ( הוא המימד החבוי של המודל) בה העמודה ה* i *הוא ווקטור הקידוד המקומי של הטוקן במקום* i*.*

*את הקידוד המקומי אפשר ליצור לפי הנוסחה (Vaswani et al., pt.3.5):*



או בשיטת קידוד מקומי נלמד (Gehring et al., pt.3.1) בה למודל של גודל רצף מקסימלי והוא לומד מטריצת קידוד מקומי בגודל וכשהמודל מקבל רצף בגודל , מטריצת הקידוד המקומי שלו היא העמודות הראשונות ממטריצת הקידוד המיקומי.

הכפלה במטריצת שיכון משוחלפת – Transposed Embedding:

(Press and Wolf)

*את התוצאה של הדיקודר אנחנו מכפילים במטריצת השיכון המשוחלפת.*

*נזכור כי הכפלה במטריצה משוחלפת היא הפעולה ההפוכה להכפלה במטריצה המקורית.*

אינטואיציה:

הדיקודר מחזיר את מטריצת שיכון של המילים במקום השני עד המקום seq len + 1 (לפני נורמליזציה).

הכפלה במטריצת שיכון היא תרגום של הסתברות לשיכון.

הכפלה במטריצת שיכון משוחלפת היא תרגות של שיכון להסתברות.

*.*

מסכת הסתקלות קדימה look ahead mask:

(Vaswani et al.)

*היא מטריצה בגודל seq\_len x seq\_len שמטרתה לגרום לכך שטוקנים לא יושפעו מהטוקנים שלפניהם.*

*כאשר seq\_len הוא אורך הרצף.*

*המסכה נוצרת לפי הקוד:*

|  |
| --- |
| def create\_look\_ahead\_mask(seq\_len: int) -> List[List[int]]:  *"""Returns a look ahead mask for the given length.  input: seq\_len: int  Returns: list of list of 0s and 1s"""* answer: List[List[int]] = [[0] \* seq\_len] \* seq\_len  for i in range(seq\_len):  for j in range(seq\_len):  if j > i:  answer[i, j] = 1  return answer |

*או לפי ההגדרה המתמטית לכל איבר:*

*מסכבת הסתכלות קדימה תמיד תיהיה מטריצה ריבועית בה האלכסון וכל האיברים מתחתיו אפס וכל האיברים מתחת לאלכסון 1.*

*דוגמה: מסכת הסתכלות קדימה לרצף באורך 3:*

*אפשר לחשוב על המסכה בטור טבלה בה האיבר בשורה j ועמודה i עונה על השאלה:*

*0 אם הטוקן במקום מושפע מהטוקן במקום ו - 1 אחרת.*

צומת לב עצמית ממוסכת בעזרת מכפלה סקלרית:

**Scaled Dot-Product** **Masked Self Attention**

(Vaswani et al.)

המטרה של צומת הלב היא לקחת ייצוג של רצף של טוקנים ולתת לכל טוקן יצוג התלוי בטוקנים שמלפניו. כל טוקן בכל שלב במודל מיוצג על ידי ווקטור באורך (קיצור של attention dimention).

נגדיר:

seq len - אורך הקלט לטרנספורמר (מספר חיובי ושלם)

*- המימד החבוי – תכונה (הייפרפרמטר) של* *המודל.*

*- מטריצות פרמטרים הנלמדים על ידי המודל.*

*הפונקציה מקבלת מטרציה*

*הפעולה מחזירה מטריצה באותו גודל של המטריצה שהיא מקבלת.*

*הגדרה מתמטית לפעולת צומת הלב:*

*אינטואיציה:*

*המצב החבוי הוא מטריצה שמכילה רצף של ווקטורים בה כל ווקטור במקום i מיצג את המשמעות הסמנטית של הטוקן במקום i בקונטקסט של המשפט.*

*אם נדמיין כל מצב חבוי של טוקן (ווקטור בגודל ) כנקודה במרחב, הכפלתו במטריצת פרמטרים תשנה את מערכת הצירים בו הווקטור נמצא למערכת צירים שמייצגת בצורה יותר מדויקת את הקשרים שבין הטוקנים השונים. בשתי מערכות הצירים מימדים.*

*השאילתה (query) של טוקן היא ווקטור שקרוב לווקטורים שיכולה להיות להם השפעה על משמעות הטוקן.*

*המפתח (key) הוא ההשפעה של הטוקן על טוקנים אחרים.*

*הערך (value) הוא התוכן של הטוקן.*

*הפעולה יוצרת מטרציה (קיצור ל dot procut – המונח לערך שסלארי באנגלית) בגודל seq len x seq len בה הוא תוצאת המכפלה הסקאלרית בין הווקטור של שמיצג את הטוקן במקום ה i בשאילתה (לאחר הטרנספורמציה) לווקטור שמיצג את הטוקן במקום ה j במפתח (לאחר הטרנספורמציה) שמייצג את ההשפעה של הטוקן במקום i על הטוקן במקום j.*

*החלוקה של כל איבר מטריצה ב היא נורמליזציה ואינה הכרחית. אם לא נחלק הפעולה תתבצע באופן דומה מאוד והמודל יעבוד בצורה מאוד דומה. הנורמליזציה משפרת קלות את ביצועי המודל.*

*נזכור שמכפלה סקאלרית בין שני ווקטורים מייצגת את הדמיון ביניהם – ככל ששני ווקטורים יותר דומים – המכפלה הסקלארית שלהם יותר גדולה ולהפך. מכפלה סקלארית יכולה להיות חיובית או שלילית. מכפלה סקלארית לא יכולה להיות יותר גדולה מאורך הווקטור הארוך יותר בריבוע.*

*הפעולה :*

*המסכה מוכפלת במינוס מיליארד ככה שכל ערך שהיה אחד במסכה המקורית הוא מינוס מיליארד במסכה המוכפלת וכל ערך שהיה אפשר נשאר אפס ובאופן פורמלי:*

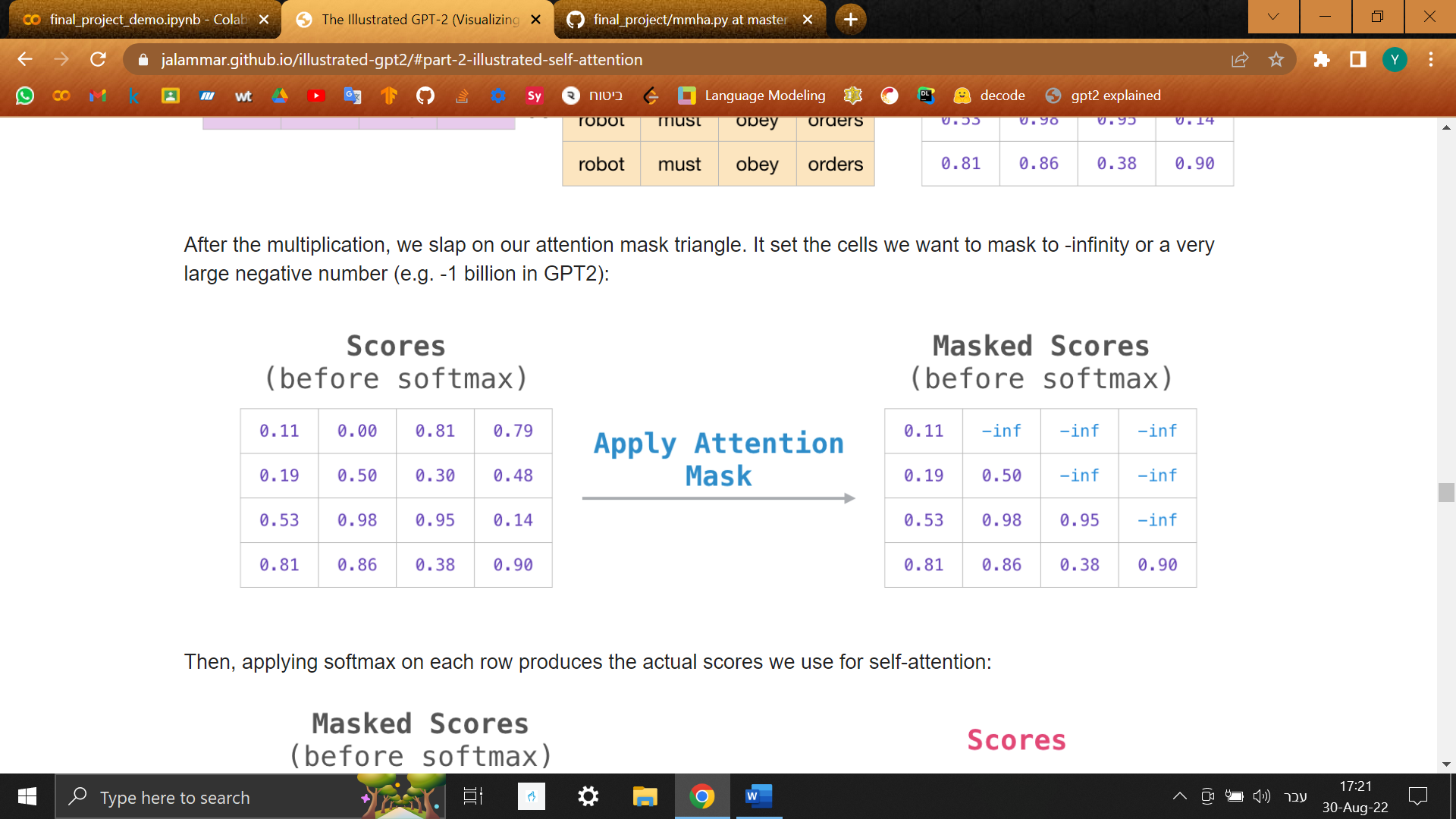
*את תוצאת המכפלה מחסרים ממטריצת הדמיון ככה שהדמיון בין*

*אינטואיציה:*

*במטריצה יש קשרים דו צדדיים בין כל הטוקנים כלומר כל טוקן משפיע על כל הטוקנים הסובבים אותו אבל זאת בעיה כי אנחנו רוצים לחזות כל טוקן מהתבסס על הטוקנים שקדמו לו בלבד.*

*אנחנו רוצים שההשפעה של טוקנים על טוקנים שבאים לפניהם ברצף תיהיה קטנה ככל הניתן.*

*המסכה גורמת להשפעה של טוקנים על הטוקנים שבאים לפניהם ברצף להיות קטנה מאוד – בערך מינוס מיליארד.*



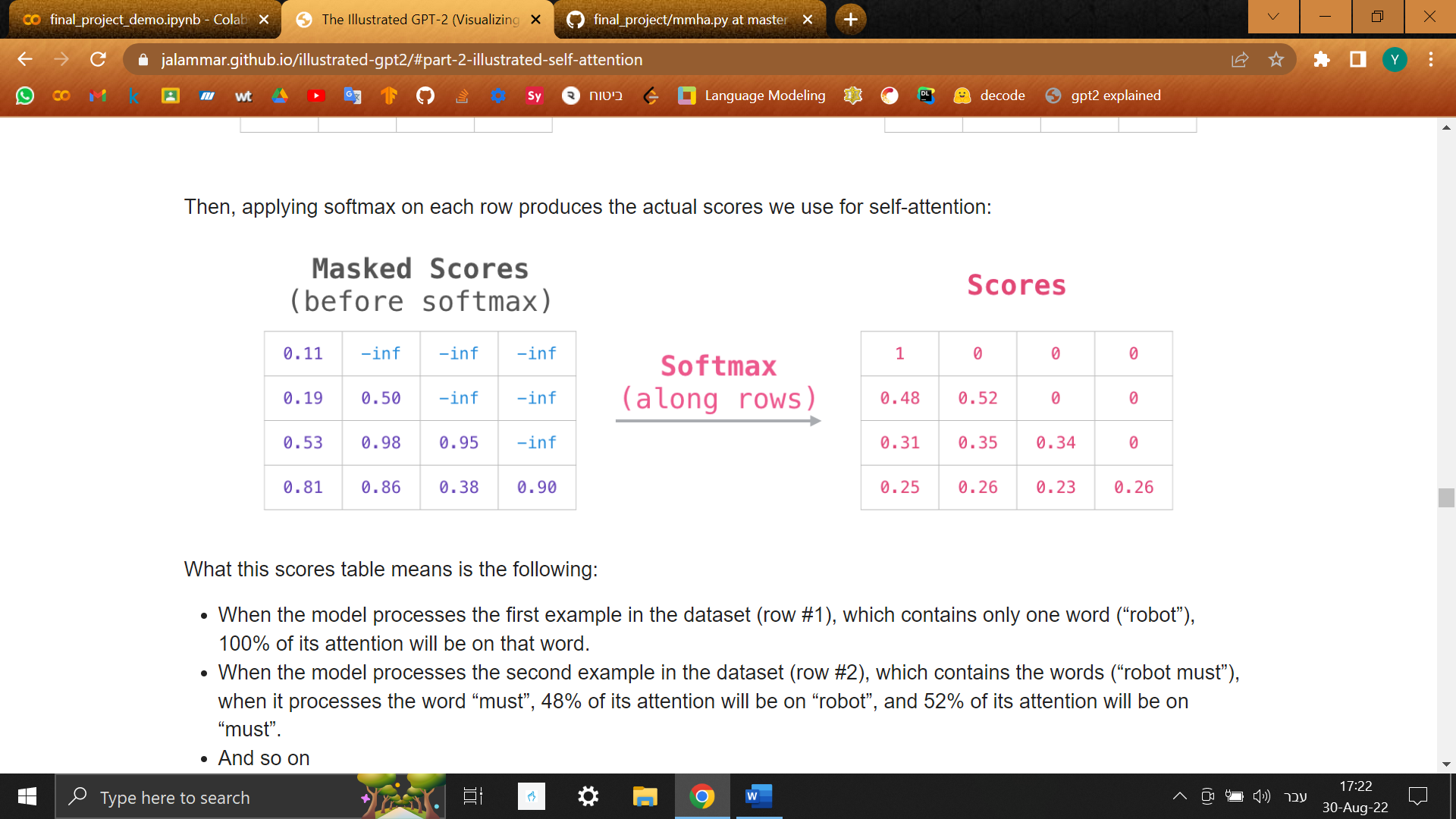
*הפעולה :*

*המטריצה ASM היא טבלה בה הוא האיבר בעמודה i ובשורה j המיצג את ההשפעה של הטוקן במקום ה i על הטוקן במקום ה j ופעולת softmax מנרמלת את הטבלה ככה שסכום כל שורה הוא 1.*

*אנחנו רוצים שההשפעה של כל טוקן על הטוקנים לפניו תיהיה קרובה מאוד ל 0.*

* *הערה: בדרך הכלל המספר הקרוב לאפס מתעגל לאפס.*

*וזאת מכיוון ש .*



*נגדיר את הערך הווקטורי של טוקן כערך שלו כפי שמיוצג במטריצה V.*

*הפעולה יוצרת מטריצת יצוג חבוי ככה ש כל טוקן מיוצג על ידי הממוצע המשוכלל של הערך הווקטורי של כל הטוקנים המשפיעים עליו.*

*אם נשתמש בדוגמה למעלה: הווקטור של הטוקן השני במטריצה יהיה 0.48 כפול הווקטור של הטוקן הראשון במטריצה ועוד 0.52 כפול הווקטור של הטוקן השני במטריצה .*

צומת לב רב-ראשית – Multi Head Attention:

(Vaswani et al.)

*בהינתן אותם פרמטרים שמקבלת פעולת צומת לב ועוד פרמטר – מספר הראשים (num\_heads)*

*(המימד החבוי של המודל חייב להתחלק ב מספר הראשים).*

*הערך החבוי האחרון והערך לפני טרנספורמציה מפוצלים ככה ש המימדים שלהם משתנים מ :*

*Seq\_len x ל num\_heads x seq\_len x ()*

*ככה שלכל ראש יש ערך חבוי אחרון וערך לפני טרנספורמציה אחרים:*

*LHV\_of\_head\_i = LHV[i, :, :]*

*PTV\_of\_head\_i = PTV[i, :, :]*

*כל ראש מחשב את צומת הלב עם הערך החבוי האחרון והערך לפני טרנספורמציה שלו ולומד פרמטרים אחרים.*

*לפיצול לראשים שתי מטרות:*

1. *להקטין את זמן החישוב:*

*החישוב של צומת לב בין מטריצות קטנות לוקח פחות זמן והחישוב של כל הראשים מתבצע במקביל.*

1. *ללמוד דברים אחרים:*

*בפיצול המטריצות כל ווקטור המתאר טוקן מפוצל ככה ש:*

*vector\_size = d\_k / num\_heads*

*start\_index = i \* vector\_size*

*end\_index = (i + 1) \** *vector\_size*

*vector\_for\_head\_i = original\_vector[start\_index:end\_index]*

*נזכור שכל איבר בווקטור מייצג תכונה סמנטית של הטוקן ולכן כל ראש מתייחס לתכונות סמנטיות אחרות של הטוקנים.*

רשת מחוברת לגמרי – Fully Connected Feed Forward Network:

*רשת מחוברת לגמרי היא שכבה שמורכבת משלוש תת שכבות:*

*שכבה דחוסה עם גודל קלט*  *וגודל פלט feed forward depth (הייפר פרמטר של המודל)*

*אקטיבציית ReLU.*

*שכבה דחוסה עם גודל קלט feed forward depth וגודל פלט .*

שכבה דחוסה\מחוברת לגמרי – Dense/Fully Connected Layer:

לשכבה דחוסה שתי תכונות*: גודל הקלט (n) וגודל הפלט (m).*

*היא לומדת מטריצת פרמטרים ו-ווקטור פרמטרים b בגודל m.*

*שכבה דחוסה מקבלת ווקטור שאורכו גודל הפלט ומבצעת עליה את הפעולה הלינארית:*

*הגדרה לכל איבר:*

*אינטואיציה:*

*נחשוב על הווקטור x בתור נקודה בתוך מערכת צירים, הכפלה של ווקטור במטריצה היא ייצוג של הווקטור במערכת צירים אחרת והוספה של הווקטור b היא הזזה של הווקטור בגודל וכיוון קבוע.*

פונקציית ReLU:

(Agarap)

*הפונקציה פועלת על כל איבר בטנזור ונוחסתה:*

*הנגזרת של הפונקציה (לפי המימוש בספריות למידת מכונה) היא:*

ריפוד - Padding:

*בלמידה עמוקה אנחנו הרבה פעמי רוצים לשלוח למודל כמה דוגמאות בו זמנית (לעשות batching).*

*עד עכשיו, הנחנו שהטרנספורמר מקבל רצף אחד אך קיימת שיטה לשלוח לטרנספורמר כמה דוגמאות בו זמנית וזאת על ידי ריפוד.*

*נוסיף לסוף כל רצף טוקן מיוחד הנקרא ריפוד ככה שכל הרצפים באותו אורך.*

*הטוקן שהמודל חוזה במקום שבו שמנו את טוקן הריפוד הוא הטוקן הבא בטקסט.*

*פונקציית המטרה נבנית ככה שטוקני ריפוד אינם חלק מהמטרה של*

*מסכת ריפוד מאפשרת את ביצוע פעולת צומת הלב על קבוצה (batch) של רצפים באורכים שונים במקביל. לכל רצף נוסיף טוקן מיוחד - ריפוד מספר פעמים בסוף הטקסט ככה שאורך כל רצף יהיה כאורך הטוקן הארוך ברצף.*

*נגדיר = האורך של רצף הארוך ביותר, = מספר הרצפים בקבוצה (באטצ').*

*ניצור טנזור בינארי לפי הכלל:*

*. אם הטוקן במקום i+1 מושפע מהטוקן במקום j וגם הטוקן במקום j הוא אינו ריפוד ו - 1 אחרת.*

*לכל רצף במקום ה i, שכבת צומת הלב משתמשת ב במקום ב וככה נוצר מצב בו ערך כל טוקן תלוי בטוקנים שלפניו* ***וגם*** *אינם טוקן הריפוד.*

**אימון ראשוני למידול שפה סיבתי בעזרת יצירת שפה:**

**Generative Pre-Training (GPT) for Causal Language Modeling**

היא שיטה לאימון מודלים שמטרתם ליצור טקסט בהינתן טקסט שמקורה במאמר (Papers with Code - GPT Explained). בה מאמנים מודל שפה סיבתי מתחיל בסט נתונים גדול עם טקסטים כלליים שעליו המודל מתאמן על מנת ליצור הבנה כללית של שפה אנושית כתובה.

בהינתן רצף של טוקנים, המודל חוזה כל טוקן בהתבסס על הטוקנים הקודמים בטקסט.

המודל אומנם מקבל את כל הטוקנים אבל בעזרת מסכת הסתכלות קדימה (look ahead mask), כל טוקן תלוי בטוקנים שבאו לפניו בטקסט בלבד.

המודל לומד לחזות איזה טוקן נמצא בכל מקום (מתוך רשימה ידועה של טוקנים).

המודל מצמם את ה categorical entropy הממוצע לרצף שאותו הוא מנסה לחזות.

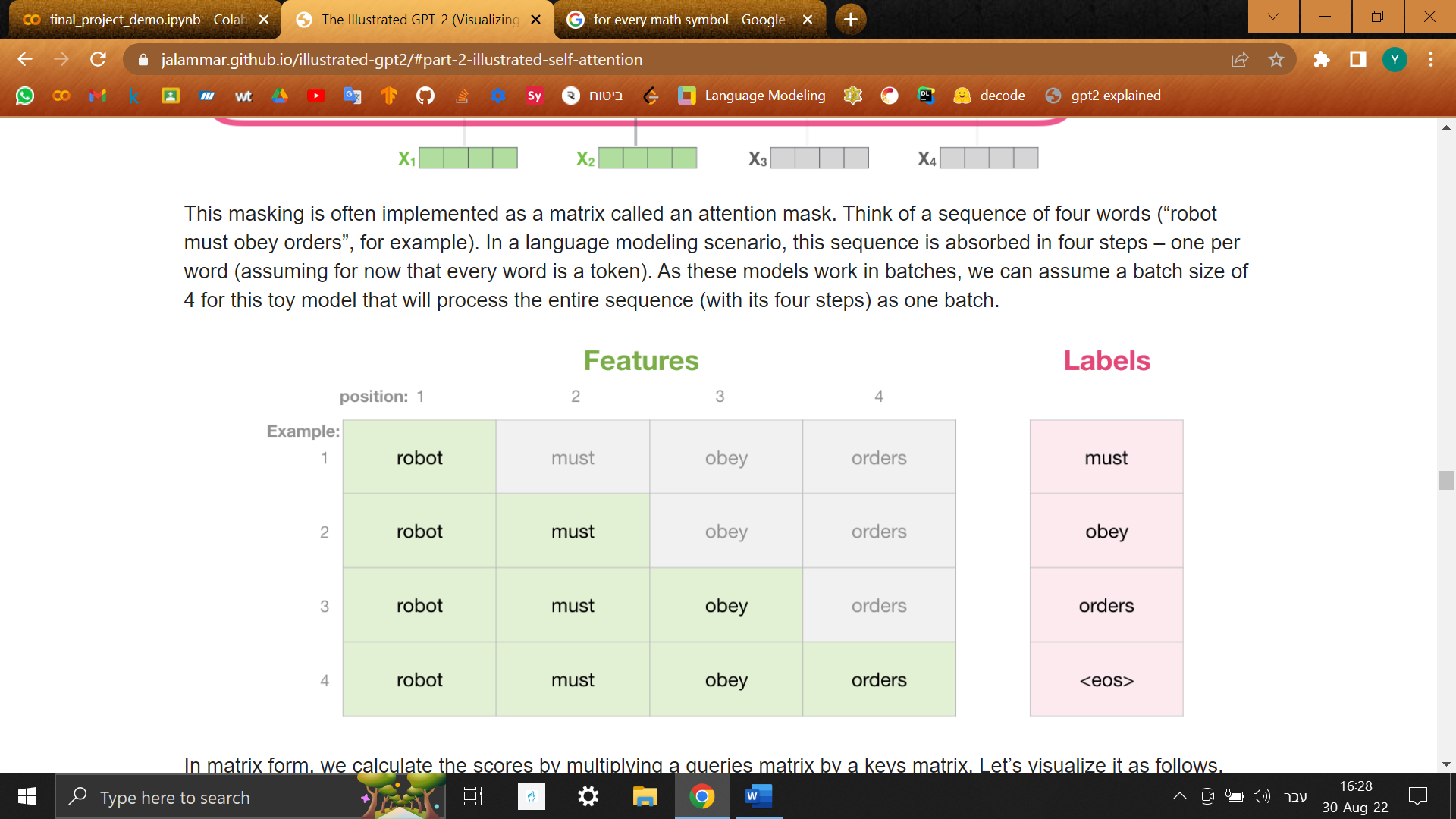
שימו לב שהמודל לא מתייחס לטוקן הראשון, זאת מכיוון שאין אפשרות לחזות את הטוקן הראשון בהתבסס על הטוקנים הקודמים (כי אין טוקנים שמגיעים לפני הטוקן הראשון).

משימה זו היא משימת לימוד בהנחיה עצמית:

המודל לומד לחזות או ליצור חלק אחד מהדוגמה בהינן חלק אחד מהדוגמה לכל דוגמה בסט הנתונים.

חלוקה של המשפט "robots must obey orders"

בכל שורה, המודל צריך לחזות מה יהיה הטוקן בתא האדום בהתבסס על הטוקנים בתאים הירוקים (הטוקנים בתאים האפורים ממוסכים).



**למידה רב שלבית – Transfer Learning:**

המשימה של חיזוי הטוקן הבא בטקסטים כלליים איננה חשובה בפני עצמה. האימון למשימת מידול שפה טבעית הוא אימון ראשוני שאחריו מגיע אימון למשימה ספציפית (למשל: סיכום טקסט, תרגום בין שפות, מענה בשירות לקוחות ועוד משימות רבות).

הרעיון מאחורי אימון כללי שלאחריו אימון למשימה ספציפית (down stream task) הוא שהמודל לומד להבין שפה כללית וליצג טקסטים באופן כללי – מה שיעזור מאוד באימון למשימות ספציפיות עם פחות נתונים ופחות כוח חישוב.

גישה זו היא סטנדרטית באקדמיה ובתעשייה בשנים האחרונות.

האימון החוזר על משימה ספציפית נעשה כמו האימון הכללי – בלי התייחסות לקלט ופלט וזאת על מנת שהמודל ילמד לעבד את המידע שמופיע בקלט.

שימוש במודל ליצירת טקסט:

בהינן טקסט שנקלט מהמשתמש:

תחילה המודל מתורגם לרצף של טוקנים על ידי הטוקנייזר.

המודל תחילה מקבל את הקלט ולאחריו טוקן רנדומלי.

המודל חוזה את הטוקן שנמצא במקום האחרון ברצף בהתבסס על הטוקנים שלפניו ולכן זאת התחזית של המודל לטוקן הראשון בפלט.

הטוקן הבא נדגם (ראה חלק על שיטות דגימה) מההסתברות שהמודל יצר.

לאחר מכן המודל מקבל את הקלט ואת הטוקן הנגדם ומוצא את הטוקן הבא וכך הלאה.

מספר הטוקנים שנוצרים יכול להיקבע בכמה דרכים:

1. על ידי המשתמש.
2. מספר הטוקנים הנוצרים יהיה שווה למספר הטוקנים בקלט.
3. על ידי מודל שמקבל קלט וחוזה את אורך הפלט.

מתוך התפלגות של הטוקנים הפלטים בסט הנתונים.

דגימה – Sampling/Decoding:

*נניח ויש לנו מודל שמקבל רצף של טוקנים ומחזירה את ווקטור ההסתברות של הטוקן הבא, יש לנו רצף של טוקנים שאנחנו רוצים שהמודל ישלים ויש לנו תנאי עצירה.*

*תנאי העצירה יכול להיות הגעה לכמות מסוימת של טוקנים או בחירה*

*דגימה היא תהליך הוספת הטוקנים לרצף. ישנן שיטות שונות לדגימה:*

דגימה לפי הסתברות מקסימלית – Argmax/Greedy Sampling:

היא שיטת הדגימה הנאיבית. בהתחילה נקבל את ווקטור ההסתברות של הטוקן הראשון על ידי הזנת הרצף המקורי למודל.

*ניקח את הטוקן שההסתברות שלו הכי גבוהה ונוסיף אותו לרצף.*

*נחזור על התהליך עד שאורך הרצף הוא האורך הרצוי.*

*יתרונות:*

* *השיטה פשוטה וקלה לישום.*
* *אין את הסיכון של לדגום טוקן עם הסתברות ממש נמוכה.*
* *אין צורך לבצע את פעולת ה softmax מכיוון שהטוקן עם ערך הלוג'יט הכי גדול בהכרח יהיה הטוקן עם ההתסברות הגבוהה ביותר.*

*החסרון:*

* *שיטה זו נוטה ליצר את הטוקנים שמופיעים הרבה בסט האימון של המודל (בדרך כלל מילות קישור) ביחס בלתי פרופורציונלי.*
* *שיטה זו נוטה לייצר טוקנים שחוזרים על עצמם.*
* *אין שליטה: אין פרמטר שאנחנו יכולים לשנות אחרי אימון המודל על מנת לשנות את הטקסט שהמודל מייצר.*

דגימה מתוך הסתברות – Pure Sampling:

*ההבדל בין דגימה מתוך הסתברות לדגימה לפי הסתברות מקסימלית היא שבדגימה רנדומלית במקום לדגום את הטוקן שהסתברותו הכי גבוהה, נדגום ברנדומליות לפי ההסתברות שהמודל חזה.*

*היתרונות:*

* *דגימה מגוונת יותר.*
* *פחות חזרתית.*

*החסרונות של השיטה:*

* *דגימה של מילים עם הסתברות נמוכה מאוד.*
* *חוסר התאמה למטרה: המודל מאומן לחזות הסתברות כמה שיותר גבוהה לטוקן הנכון ובפונקציית המטרה אין התייחסות להסתברות של הטוקנים האחרים.*
* *תלות ברכיב רנדומלי.*

דגימה מתוך k הטוקנים שהסתברותם הכי גבוה – Top k Sampling:

*(Fan et al. 5)*

*נבחר מספר שלם וחיובי k קטן או שווה לכמות הטוקנים שהמודל מכיר.*

*בהינתן ווקטור הסתברות:*

*נמצא את k הטוקנים שהתברותם הגדולה ביותר.*

*נקבע את ההסתברות של שאר הטוקנים לאפס.*

*נחלק את ההסתברות של כל טוקן בסכום של הווקטור החדש (על מנת לקבל ווקטור שסכומו אחד).*

*ונדגום מהווקטור שנוצר.*

*יתרונות של השיטה:*

*ברוב המקרים, כל ההסתברויות השונות מ – 0.*

*יתרונות:*

* *נותנת חסם מלמטה להסתברות של הטוקנים האופן שתלוי בווקטור ההסתברות.*
* *מאפשרת דגימה באופן שמתייחס לטוקנים שהסתברותם גבוהה בלבד.*

*חסרונות:*

* *לא מונעת לגמרי את האפשרות לדגום טוקנים שהסתברותם נמוכה.*
* *מגדירה טוקן עם הסתברות גבוהה כטוקן שיש פחות מ k טוקנים שהסתברותם גבוהה מהסתברותו – הגדרה שיכולה ליצור בעיות במקרי קצה.*
* *תלות ברכיב רנדומלי.*

*הערה: דגימה מתוך k הטוקנים שהסתברותם הכי גבוה כאשר k שווה אחד היא דגימה לפי הסתברות מקסימלית.*

דגימה מתוך הטוקנים שסכום הסתברותם <= p – Top p Sampling:

*(Holtzman et al.)*

*נבחר מספר p בין אפס ואחד.*

*ניצור ווקטור הסתברות חדש בו כל ההסתברויות 0.*

*ונעקוב אחרי סכום ההסתברויות.*

*נעבור על ווקטור ההסתברויות המקורי לפי סדר:*

*נעבור על הטוקנים לפי סדר ההתסברות כל עוד סכום ההתסברויות קטן מ p:*

*בכל פעם נוסיף לווקטור החדש את הטוקן הנוכחי בצורה הבאה:*

|  |
| --- |
| new\_probs[token\_id] = curr\_token\_prob.prob |

*ונוסיף את ההסתברות לסכום ההסתברויות*

*נעבור על הווקטור ונחלק את ההסתברות של כל טוקן בסכום ההסתברויות (על מנת לקבל ווקטור שסכומו 1).*

*ונדגום מהווקטור שנוצר באופן רנדומלי.*

*יתרונות (לעומת דגימה מתוך k הטוקנים שהסתברותם הכי גבוה):*

* *ההגדרה של הסתברות גבוהה יותר עמידה בפני מקרי קצה.*
* *השליטה בבחירת הטוקנים יותר טובה.*

*חסרון:*

* *אין חסם המנוע בחירת טוקנים בעלי הסתברות נמוכה.*
* *תלות ברכיב רנדומלי.*

חיפוש עץ – Tree Sampling:

*נגדיר את ההסתברות של רצף T בו l טוקנים :*

*נבחר מספר חיובי שלם קטן או שווה לכמות הטוקנים שהמודל מכיר ונקרא לו רוחב העץ .*

*בכל שלב בחיפוש, נבחר את (רוחב העץ) הטוקנים שהסתברותם הכי גבוהה תוך מעקב על ההסתברות של הרצף.*

*לכל טוקן שבחרנו, נחזבחר את (רוחב העץ) הטוקנים שהסתברותם הכי גבוהה תוך .*

*נחזור על התהליך עד שנגיע למספר הטוקנים הרצוי.*

*לאחר מכן נבחר את הרצף שהסתברותו הגבוהה ביותר.*

*חסרון:*

*סיבוכיות זמן ריצה גדולה כשמייצרים רצפים ארוכים.*

*כאשר n הוא מספר הטוקנים שאנחנו רוצים לחזות.*

*אנחנו בעצם יוצרים עץ בו לכל הורה w ילדים ויצירה של ילדים נעשית באמצעות קיראה למודל ולכן אנחנו קוראים למודל פעמים וזמן הריצה של מודל מסוג טרנספורמר הוא ולכן זמן הריצה של הדגימה הוא בדיוק זהו זמן ריצה גדול יותר מזמן ריצה אקספנונציאלי.*

חיפוש עץ עם אילוץ על סכום ההסתברות:

*כמו חיפוש עץ רק שבמקום לבחור את w הטוקנים שהסתברותם הכי גדולה, בוחרים את הטוקנים שהסתברותם הכי גדולה ככה שסכום ההסתברויות לא עולה על p (בדומה ל דגימה מתוך הטוקנים שסכום הסתברותם <= p)****.***

*אם המודל חוזה הסתברות שווה לכל טוקן, יבחרו טוקנים כאשר v הוא מספר הטוקנים שהמודל מכיר ולכן פונקציית זמן הריצה היא ולכן זמן הריצה הוא אקספוננציאלי.*

דגימה עם עונשים - penalized sampling:

(Keskar et al.)

על מנת למנוע טקסט שחוזר על עצמו, דגימה עם עונשים מקטינה את הלוג'יט (וכתוצאה מכך את הההסתברות) של טוקנים שהופיעו כבר ברצף שהטוקנים שהמודל יצר פי **θ** כאשר **θ** פרמטר גדול מ 1.

מדד ברט – BERT Score:

(Zhang et al.)

*מדד ברט מודד את הקרבה הסמנטית בין טקסט אחד מועמד לטקסט מטרה בעזרת מודל שפה מסוג BERT – מודל השפה מקבל טקסט ונותן לכל טוקן ווקטור שמייצג את המשמעות של הטוקן בתוך הקוטקסט של שאר המילים במשפט.*

*על מנת ליצור את מדד ברט, יוצרים לכל טקסט ייצוג כרצף ווקטורים.*

*לכל טוקן בטקסט המועמד מוצאים את הטוקן הכי קרוב עליו סמנטית בעזרת מכפלה סקלארית בין הווקטורים המייצגים את הטוקנים (הווקטורים מנורמלים כך שהמכפלה הסקלארית ביניהם שקולה לדמיון קוסינוס שלהם). והתוצאה של הטוקן היא המכפלה הסקלארית שלו עם הטוקן הכי קרוב. את התוצאות של הטוקנים סוכמים לסקלאר.*

*כאשר x הוא המטריצה המתארת את טקסט המטרה ו y הוא הטקסט המועמד הנוסחה היא:*

***b*** *הוא קו ההתחלה (baseline) שערכו הוא מדד ברט (precision recall or F1 בהתאמה) הממוצע לזוגות משפטים רנדומלים מסט נתונים גדול.*

*השקילה למדד נתון שנקרא לו PS הוא:*

*המדדים שאציג בתוצאות הם לאחר שקילה.*

*לא השתמשתי באפשרות לשקילה לפי תפוצת הטוקנים ולכן לא אפרט עליה.*

*עברית: זהו משפט לדוגמה.*

*אנגלית: "*

סקירת פייתון

בחלק זה אסקור נושאים מתקדמים בתכנות ונושאים בשפת פייתון בהם השתמשתי בעבודה.

אני לא אסקור נושאים בסיסיים (כגון ירושה וייבוא) מכיוון שהם אינם נושא הפרויקט.

רשימת מבני נתונים בפייתון ומבני הנתונים המקבילים בשפות אחרות:

רשימה (list) – מערך דינמי. מושגים מקבילים: dynamic array, array list.

טאפל (tuple) – מערך שאינו ניתן לעריכה. מושג מקביל: imutable array.

מילון (dict) – מפה\מפת גיבוב. מושגים מקבילים: map, hash map.

סט (set) – קבוצה – רשימה בה אף איבר לא חוזר על עצמו. מושג מקביל: hash set.

רמזי סוג – Type Hints:

(van Rossum et al.)

מערכת הטיפוסים בפייתון היא דינמית כלומר הטיפוסים בפייתון נקבעים בזמן הריצה ולא בזמן כתיבת התוכנית או הקומפילציה.

פעמים רבות נרצה לדעת מה הטיפוס של ביטוי משתנה או פונקציה בזמן קריאת הקוד ולכן יש צורך ברמזי סוג.

רמזי סוג הם הדרך המקובלת להוסיף מידע על סוג של נתונים (במקום הערות).

חשוב לציין כי רמזי סוג אינם מחייבים וכי השמת ערך במשתנה כאשר סוג הערך שונה מהרמז לסוג המשתנה אינו גורר שגיאה.

הספרייה הסטנדרטית typing מכילה כלי עזר לרמזי סוג.

רב צורתיות (פולימורפיזם):

עיקרון במדעי המחשב לפיו יש לקרוא למתודות אשר מבצעות את אותו תפקיד בשם זהה.

לא משנה איזה טיפוס נעביר לפונקציה, היא עושה את ההתאמות הנדרשות.

לדוגמה: from\_dict, as\_dict, \_\_init\_\_, len, str…

פעולות קסם – Magic/Dunder Methods:

פעולות קסם הן פעולות שמורות בשפת פייתון שיש להן תפקיד מיוחד:

פונקציות קסם ממשות את עקרון הרב צורתיות – מחלקות רבות ממשות כל פונקצייית קסם.

דוגמה לכך היא הפונקציה str שהופכת כל עצם למחרוזת, בלי קשר לסוגו.

לדוגמה: הפונקציה הבונה\מאתחלת - \_\_init\_\_ שנקראית כשפותחים סוגריים בצמוד לשם המחלקה, \_\_call\_\_ שנקראית כשפותחים סוגריים בצמוד לשם של משתנה,

\_\_del\_\_ שנקראית כשמוחקים עצם באמצעות המילה השמורה delete, ועוד רבות.

קשטנים - :Decorators

*(Smith)*

קשטן היא פעולה שמקבלת פונקציה או מחלקה ומחיזרה פונקציה או מחלקה חדשה בהתאם.

הקשטן בדרך כלל מוסיף פונקציונליות חדשה לפונקציה או מחלקה בלי לשנות את הלוגיקה.

דוגמה: קשטן שמדפיס את הטיעונים בכל קריאה לפונקציה:

|  |
| --- |
| def printer(original\_function: Callable) -> Callable:  *"""A decorator that prints  the arguments of   the function it decorates  every time it is called."""* def new\_function(args):  print(args)  return original\_function(args)   return new\_function |

הסינטקסט לשימוש בקשטן:

|  |
| --- |
| @my\_decorator def original\_function():  # original function's code |

מקביל לסינטקס:

|  |
| --- |
| original\_function = my\_decorator(original\_function) |

מחלקת בסיס אבסטרקטית:

(Guido van Rossum and Talin)

מחלקות אבסטרקטיות הינן מחלקות שלא ניתן לייצר מהן אובייקטים ומטרתן היחידה היא להוות מחלקת בסיס. במקרים בהם יש צורך להגדיר מחלקות בסיס לאובייקטים ממשיים ניתן להגדירם כמחלקות אבסטרקטיות.

לדוגמה: ניצור מחלקה אבסרקטית המייצגת צורה וממנה יורשות המחלקות ריבוע, משולש עיגול...

אנחנו רוצים ליצור עצמים ממחלקות המשנה (למשל ריבוע) ורוצים למנוע יצירת צורה כללית שאיננה אחת מהצורות הספציפיות (ריבוע, עיגול, משולש...).

על מנת ליצור מחלקה אבסטרקטית, אשתמש בקשטן abc.ABC.

פונקציה אבסרקטית:

(Guido van Rossum and Talin)

היא פונקציית מחלקה (mathod) של מחלקת בסיס אבסטרקטית שממומשת בנפרד לכל אחת מהמחלקות היורשות.

לדוגמה – במחלקה המייצגת צורה דו מימדית נרצה שלכל אחת מתת המחלקות תיהיה פונקציה המחשבת שטח ובעזרת נוחסה שונה לכל תת מחלקה. לשם כך ניצור פונקציה בשם calculate area שלא מקבלת פרמטרים ומחזירה את השטח במטרים רבועים כמספר עשרוני.

על מנת ליצור פונקציה אבסטרקטית, אשתמש בקשטן abc.abstract\_method

פעולה סטטית:

(Rossum, pt.2)

פעולה סטטית היא פעולה ששייכת למחלקה עצמה ולא לעצם.

אחד השימושים הנפוצים של פעולות סטטיות הוא כפעולות עזר לפעולות לא סטטיות.

לדוגמה – במחלקה המייצגת רובוט שזורק חפצים, נרצה לממש פונקציית עזר שמקבלת מיקום של הרובוט, מסת החפץ הנזרק ומיקום רצוי של החפץ ומחשבת את זווית ומהירות הזריקה.

זוהי איננה פונקציה פנימית מכיוון שהיא לא תלויה בתכונות של הרובוט אך מקומה כן בתוך האובייקט מכיוון שהיא ממשת פעולה שקשורה למהות המחלקה.  
פונקציות בונות הן פונקציות סטטיות.

על מנת ליצור פעולה סטטית, אשתמש בקשטן staticmethod.

מחלקת נתונים – Data Class:

*(Eric V. Smith)*

הקשטן dataclasses.dataclass יוצר מחלקה ליצוג נתונים בעזרת שמות ורמזי הסוג של משתני המחלקה במחלקה שהוא מקבל.

הקשטן יוצר פונקציות כגון: אתחול - \_\_init\_\_, יצוג כמחזורת - \_\_str\_\_ ובדיקת שוויון: \_\_eq\_\_.

למחלקת נתונים אפשר להוסיף פונקציות ואף לדרוס את הפונקציות שנוצרות על ידי הקשטן.

המחלקה האבסרקטית Callable:

(Guido van Rossum and Talin)

היא מחלקה המגדירה עצם קריא.

עצם קריא הוא עצם בעל פונקציית הקסם \_\_call\_\_ . הסינטקס:

My\_callable(arg1, key\_word=arg2)

מקביל לסינטקס:

My\_callable.\_\_call\_\_(arg1, key\_word=arg2)

הערה: פונקציות ועצמים קריאים הם מונחים מקבילים – פונקציות הן עצמים קריאים ועצמים הם פונקציות קריאות.

Enum:

(Warsaw et al., 2013)

מחלקה המייצגת סט קבוע של ערכים בעלי שם שנקבעים על ידי המתכנת.

ערימת מקסימום\מינימום – Maximum/Minimum Heap:

(‘Heapq — Heap Queue Algorithm’)

ערימת מינימום\מקסימום היא עץ בינארי כמעט שלם בו כל אב קטן (בערימת מינימום)\גדול (בערימת מקסימום) מבניו. כתוצאה מכך, השורש הוא החוליה בעלת הערך הקטן (בערימת מינימום)\גדול (בערימת מקסימום) ביותר בעץ וניתן להוציא אותה מהעץ בסיבוכיות זמן  ניתן להפוך רשימה לערימת מינימום בסיבוכיות זמן ריצה .

בעזרת ערימה ניתן למצוא את k הערכים בעלי הערכים הקטנים\גדולים ביותר ברשימה בסיבוכיות זמן ריצה .

פיתוח מודלי למידה עמוקה בעזרת tensorflow:

הספרייה tensorflow מכילה כלים רבים ליצירה ואימון של מודלי למידה עמוקה באופן אמין ופשוט.

שניים מהכלים הם יצירת שכבות ומודלים בעזרת יצירת מחלקות היורשות מהמחלקה Layer ו Model בהתאם.

המחלקה Module:

נשתמש בה כשנרצה ליצור מחלקה היורשת מהמחלקה Callable שעובדים בצורה אופטימלית עם טנזורים. מחלקה היורשת באופן ישיר מ Module לא אמורה להשתמש במחלקות אחרות שיורשות באופן ישיר מ Module אחרים אלא רק בפונקציות הבנויות בספרייה.

היא מכילה אופטימיזציות רבות לפעולות על טנזורים.

הפונקציה call היא הפונקציה שנקראית באופן עקיף כשקוראים לאובייקט שמממש את המחלקה Module ובתוכה נכתוב את הלוגיקה של המחלקה.

המחלקה Layer:

יורשת מהמחלקה Module. נירש ממנה ישירות על מנת ליצור מחלקה המשתמשת במודולים ושכבות אחרות.

שכבות בנויות מראש:

בספרייה מומשות שכבות שימושיות כגון שכבה דחוסה, שיכון ורגורליזציה.

כל שכבה מומשה על ידי ירישה מהמחלקה Layer.

המחלקה Model:

הפונקציה compile מקבלת מודול והופכת אותו לפונקציה בשפת c++ ולאחר מכן מעבירה אותו קומפילציה.

הפונקציה fit של המחלקה מקבלת זוגות של קלט ופלט ומאמנת את המודל בגישת אימון מונחה ולכן לא אוכל להשתמש בה למידול שפה בגישת לימוד בהנחיה עצמית.

במידה ויש לנו מודול שמורכב מכמה שכבות\מודלים שונים, קריאה לפונקציה complie קוראת לפונקציה compile של תת השכבות והמודלים.

פיתוח התכונה

בחלק זה של העבודה אעסוק בפירוט בתכונה שכתבתי ואסביר את חלקי הקוד ואת ההחלטות שלקחתי בכתיבת התוכנה.

האלגוריתם ליצירת טקסט בעזרת קבוצות של טוקנים:

ביצירת טקסט בקבוצות, יצירת הטקסט נעשית כמו ביצירת טקסט "רגילה" רק שבמקום שהאלגוריתם יחשב את ההסתברות של הטוקן הבא – האו מחשב את ההסתברות של כמה מהטוקנים הבאים בהתבסס על הטוקנים שהוא כבר יצר.

**יש לקרוא את החלק התיאורטי לפני שקוראים את ההסבר על מימוש האלגוריתם.**

הקוד של האלגוריתם נמצא בקבצים: text\_generator.py, sampling\_generator.py, tree\_generator.py. הקבצים מופיעים כנספח בקובץ זה.

המחלקה :TextGenerator

היא מחלקת בסיס אבסטרקטית.

המחלקה מייצגת אובייקט קריא (callable) שטוען מודל שפה סיבתי מ – [hugging face hub](https://huggingface.co/models?pipeline_tag=text-generation&sort=likes) ומשתמש בו ליצירת טקסט מתוך טקסט שנקלט מהמשתמש בעזרת שיטת הדגימה שפיתחתי.

משתני המחלקה:

שם מודל, מודל, טוקנייזר, גודל קבוצה, טמפרטורה, מכפיל אורך תשובה, גודל אוצר מילים (ישנם עוד משתני מחלקה אך הם אינם חשובים למהות המחלקה).

מכפיל אורך התשובה משמש כחסם מקסימלי לאורך הטקסטים שהאלגוריתם מייצר. האלגוריתם יכול ליצור טקסטים שאורכם רטן או שווה לעיגול למטה של מכפלת אורך הפרומפט במכפיל אורך התשובה.

בחרתי להוסיף חסם מקסימלי לאורך הטקסט הנוצר מכיוון ששמתי לב שהאלגוריתם מייצר טקסטים ארוכים במיוחד ושיערתי שזה יפגע ביכולת למדוד את איכות הטסטים בעזרת מדד ברט.

הפונקציה \_\_call\_\_ מקבלת פרומפט אחד או יותר כמחרוזת, תוספות (תחילית וסופית) לפרומפט, מספר ההשלמות השונות לכל פרומפט, האורך המקסימלי של כל השלמה, אסטרטגיית קיטועת ופרמטרים בוליאנים שקובעים: האם להחזיר את ההשלמה כמחרוזת והאם להחזיר אותה כרצף טוקנים כרצף טוקנים והאם לחבר טוקנים שונים למילה אחת.

הפונקציה: create\_prob\_matמקבלת קלט למודל בצורת מחרוזת או רצף טוקנים.

ומחזירה רשימה דו מימדית (מטריצת ההסתברות) בה prob\_mat[a][b] היא ההסתברות של הטוקן שהמזהה שלו b להיות האיבר ה a בהמשך של הטקסט הנקלט.

זאת על ידי הוספה של (גודל הקבוצה פחות 1) טוקני ריפוד בסוף הרצף.

אינטואיציה:

LHM – look ahead mask – מסכת הסתכלות קדימה.

יצירת הטוקן הבא הבא (בקבוצות של טוקן אחד):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| next token: |  |  |  |  |  |
| טוב | [LHM] | [LHM] | [LHM] | [LHM] | בוקר |
| לכולם | [LHM] | [LHM] | [LHM] | טוב | בוקר |
| , | [LHM] | [LHM] | לכולם | טוב | בוקר |
| היום | [LHM] | , | לכולם | טוב | בוקר |
| אנחנו | היום | , | לכולם | טוב | בוקר |

יצירת שני הטוקנים הבאים (בקבוצות של שני טוקנים):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Next token |  |  |  |  |  |
| לכולם | [LHM] | [LHM] | [LHM] | טוב | בוקר |
| , | [LHM] | [LHM] | [PAD] | טוב | בוקר |
| היום | [LHM] | , | לכולם | טוב | בוקר |
| אנחנו | [PAD] | , | לכולם | טוב | בוקר |

דוגמה: אם הקלט הוא המילה "בוקר" ומזהה המילה "טוב" הוא 95 וההסתברות (לפי המודל) שהמילה "טוב" תופיע מיד אחרי המילה "בוקר" הוא 80% אז prob\_mat[0][95] יהיה 0.8 ואם מזהה המילה "לכולם" הוא 117 וההסתברות (לפי המודל) שהמילה "כולם" תופיע בצורה "בוקר" (מילה לא ידועה) "לכולם" היא 60% אז prob\_mat[1][117] יהיה 0.6.

סיבכויות זמן הריצה של הפונקציה היא  כאשר הוא מספר הטוקנים ברצף שהפונקצייה מקבלת ו הוא גודל הקבוצה.

זאת משום שסיבוכיות זמן הריצה של קריאה למודל שפה סיבתי היא  *וגודל הקלט שהפונקציה מכניסה למודל השפה הסיבתי הוא וגודל הקלט בריבוע הוא*

*וידוע כי:*

*ולכן*

הפונקציה get\_token\_tensor היא פונקציית עזר של preprocess ופונקציית מעטפת לפונקצייה \_\_call\_\_ של הטוקנייזר שהופכת מחרוזת לטנזור של שלמים המכיל רצף של מזהיי טוקנים.

הפקנצייה preprocess מקבלת את הפרומפט, תוספות לפרומפט (תחילית וסופית) ואסטרטגיית קיטוע. היא מחזירה רצף טוקנים שמכיל את הטוקנים של הפרומפט והתוספות ואת אורך (מספר הטוקנים) הפרומפט והתוספות.

הפונקציה האבסרקטית \_forward מקבלת רצף של הטוקנים בפרומפט ובתוספות (הרצף שנוצר בפונקציה preprocess), מספר הטוקנים המקסימלי ליצור ומספר הרצפים השונים ומחזירה רצף\רצפים של טוקנים הכולל את הטוקנים מהפרומפט והתוספות ואת הטוקנים שהאלגוריתם בחר.

הפונקציה postproccess מקבלת: את רצף הטוקנים שנוצר על ידי \_forward, המספר המקסימלי של הטוקנים הנוצרים, גודל הפרומפט והתוספות, האם להחזיר את הטקסט כמחרוזת, האם להחזיר את הטקסט כרצף טוקנים, האם להשאיר את התוספות כחלק מההשלמה, והאם לחבר טוקנים שונים למילה אחת.

הפונקציה מחזירה את הטקסט שנוצר על ידי האלגוריתם כמילון עם המפתחות:

"generated\_text", " generated\_token\_ids"

המחלקה TreeGenerator:

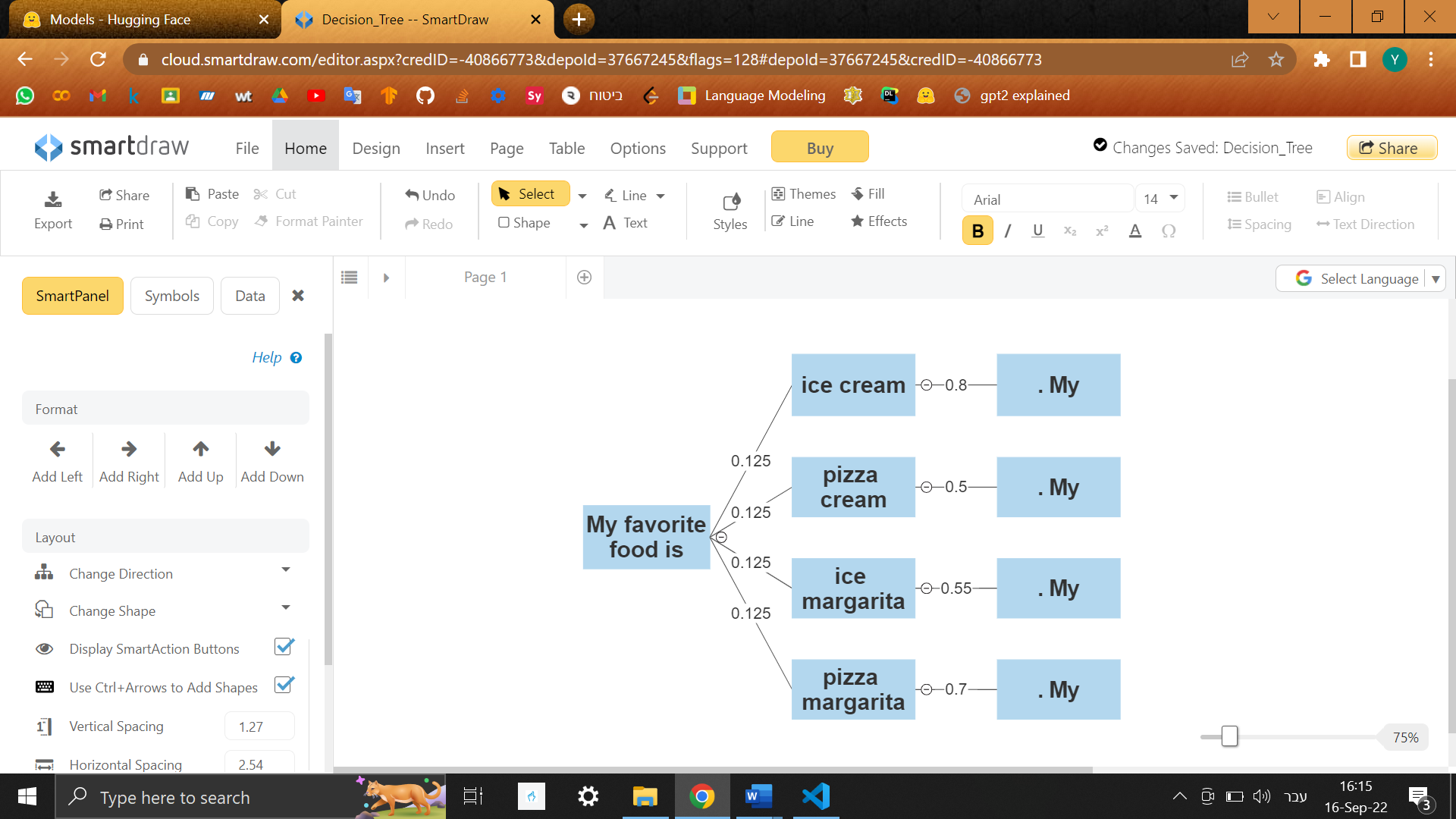
יורשת מהמחלקה TextGenerator ומייצרת טקסט בעזרת חיפוש עץ בקבוצות:

ההסתברות של קבוצה מוגדרת במכפלת הסתברות הטוקנים בה. והסתברות של טקסט מוגדר כמפלת הסתברות הקבוצות.

דוגמה להמחשה – דגימה של 4 טוקנים כאשר גודל הקבוצה הוא 2 ו top\_pהוא 0.5 עבור הפורמפט "My favorite food is":

אם המזהה של הטוקנים pizza, ice, cream, margarita הם המספרים אפס עד שלוש בהתאם, מטריצת ההסתברות של הקבוצה הראשונה תיראה באופן הבא:

ובחירת הקבוצות תעשה באופן הבא:



הטקסט שיווצר הוא: "My favorite food is ice cream. My" מכיוון שהסתברותו הכי גבוהה.

המחלקה SampleGenerator:

יורשת מהמחלקה TextGenerator ומייצרת טקסט בעזרת דגימה בקבוצות.

היא מתחילה לולאה כאשר הרצף הנוכחי הוא הטוקנים של הפרומפט.

בכל חזרה היא יוצרת מטריצה בצורה [גודל הקבוצה, גודל המילון], מכל ווקטור נדגם טוקן והוא נוסף לסוף הרצף. הלולאה ממשיכה עד הגעה לטוקן מיוחד שמעיד על סוף המשפט או עד שנוצר מספר מסויים של טוקנים.

בהתאם לפרמפטרים של הג'טנרטור, הוא משתמש באחת מהפונקציות הבאות על מנת לדגום טוקן מווקטור הסתברות:

הפונקציה unfiltered\_sampling: משתמשת בדגימה מתוך הסתברות.

הפונקציה highest\_prob\_token: משתמשת בדגימה

הפונקציה :top\_k\_sampling דגימה מתוך k הטוקנים שהסתברותם הכי גבוה – Top k Sampling כאשר k הוא המשתנה .top\_k

הפונקציה :top\_p\_sampling משתמשת דגימה מתוך הטוקנים שסכום הסתברותם <= top\_p.

לדוגמה:

דגימה של 4 טוקנים כאשר גודל הקבוצה הוא 2 ו top\_pהוא 0.49 עבור הפורמפט

"My favorite food is":

אם המזהה של הטוקנים pizza, ice, cream, margarita הם המספרים אפס עד שלוש בהתאם, מטריצת ההסתברות של הקבוצה הראשונה תיראה באופן הבא:

נגריל את הטוקן הראשון לפי השורה הראשונה במטריצה ונקבל את הטוקן "pizza"

נקבע את ההסתברות של הטוקן "pizza" להיות אפס ונחלק כל איבר בווקטור ההסתברות בסכום הווקטור ונקבל שווקטור ההסתברות של הטוקן השני הוא:

*ואז נגריל את הטוקן "margarita".*

*לאחר מכן נתחיל את אותו התהליך עם הטקסט "My favorite food is pizza margarita" וככה הלאה...*

אפליקציית הווב:

באפליקציית הווב המשתמשים יכולים להשתמש באלגוריתם לדגימה בקבוצות בעזרת מודלים מהאתר hugging face hub עם פרמטרים שונים במטרה להשוות בין שיטות דגימה, בין מודלים ובין פרמטרים של הדגימה (גודל הקבוצה, טמפרטורה, top\_p, top\_k...).

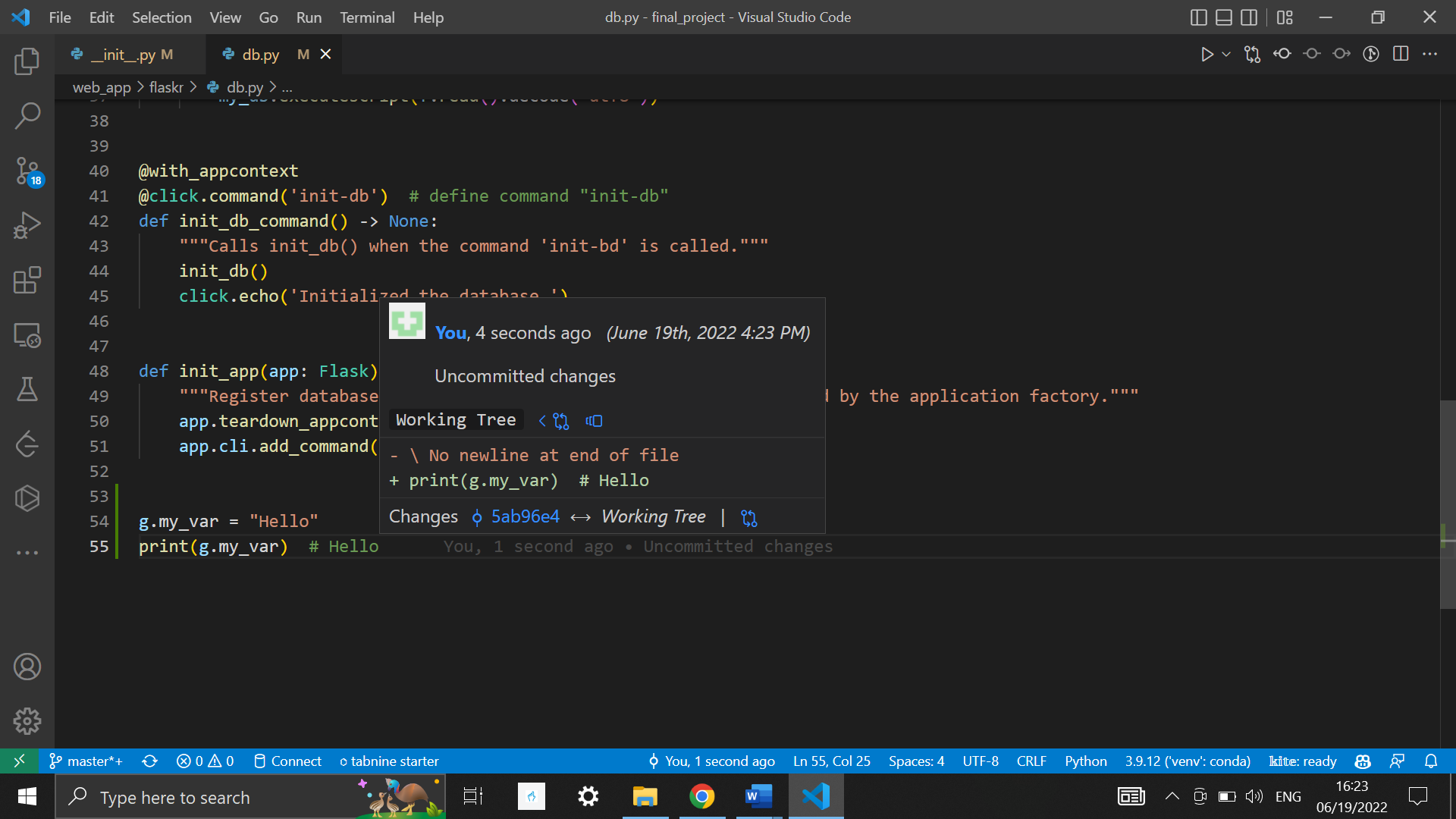
כל ההשלמות נשמרות בבסיס נתונים וכל משתמש יכול לראות את התוצאות של כל שאר המשתמשים.

המחלקה Blueprint:

נועדה על מנת לחלק אפליקציית Flask למספר חלקים, כל חלק תחת תת כתובת (subdomain) אחרת.

המשתנה הגלובלי :g

מוגדר כשהמשתמש נכנס לאפליקציה. הוא ריק כל עוד לא מכניסים אליו שום דבר. אפשר לשמור בו משתנים (מכל סוג) לפי שם ולגשת בו לכל משתנה לפי שם:



**הקובץ \_\_init\_\_.py:**

כל אפליקציה שמפותחת באמצעות Flask חייבת לכלול קובץ ששמו: "\_\_init\_\_.py" ובו פונקציה הנקראית "create\_app" בלבד. פונקציה זאת נקראת כשהשרת מתחיל להריץ את האפליקציה. הפונקציה מבצעת את הפעולות הבאות: יצירת עצם האפליקציה מהסוג Flask, הגדרת קונפיגורציה (למשל מיקום בסיס הנתונים), יצירת תקייה לבסיס הנתונים, יצירת בסיס הנותנים ושמירת blueprints.

הקובץ database.py:

בקובץ database.py נמצאות הפונקציות שאחראיות על ניהול בסיס הנתונים:

הפונקציה get\_db() בודקת אם קיים חיבור לבסיס הנתונים (עצם מהמחלקה sqlite3.Connection) במשתנה הגלובלי ואם לא, יוצרת אחד כזה ושומרת אותו ב g.my\_db ולאחר מכן (בלי קשר לתנאי הראשון) מחזירה את g.my\_db.

הפונקציה init\_db() מקבלת חיבור למסד הנתונים ומפעילה את פקודות ה SQL שבקובץ schema.sql.

הקובץ schema.sql:

מכיל את פקודות ה sql הבאות:

אם קיימות טבלאות בשמות: completion, model, user מחק אותן.

צור את הטבלאות הבאות (כל הטבלאות בבסיס הנתונים):

**user המאחסנת משתמשים עם העמודות:**

id - מספר סידורי: שלם של המשתמש שהוא המפתח הראשי של הטבלה.

username – שם משתמש: טקסט ומיוחד.

password – סיסמה מוצפנת: טקסט (הסיסמה מוצפנת לפני שהיא נכנסת לבסיס הנתונים).

**model המאחסנת מודלים עם העמודות:**

Id - מספר סידורי של המודל שהוא המפתח הראשי של הטבלה.

user\_id – מספר סידורי של המשתמש הראשון שהשתמש במודל. (בין מודל למשתמש יש קשר רבים לרבים).

model\_name – שם הקובץ של המודל כפי שמופיע ב hugging face hub.

Created – הזמן בו השתמשו לראשונה במודל.

**completions – המאכסנת השלמות עם העמודות:**

id – מזהה ההשלמה.

User\_id - מספר משתמש.

model\_id – מספר מודל.

created – הזמן בו נוצרה ההשלמה.

prompt – הקלט לאלגוריתם.

answer - הפלט של האלגוריתם.

num\_tokens – מספר הטוקנים שיוצרו על ידי האלגוריתם.

Generation type – איזה אלגוריתם יצר את הטקסט?

top\_p, top\_k, temperature – פרמטרים של האלגוריתם.

הקובץ auth.py:

מכיל את ה blueprint auth שכולל את הפונקציה login ו register שמציגות את עמודי הכניסה והרשמה לאתר. כל אחת מהפונקציות קולטת את שם המשתמש והסיסמה מעמוד ה HTML. אם המשתמש נרשם בהצלחה הוא מועבר לעמוד ההתחברות ואם הוא התחבר בהצלחה הוא מועבר לעמוד completion.index.

הפונקציה logout מנקה את הסשין ומעבירה את המשתמש לעמוד completion.index.

הפונקציות login, logout, register מופעלות (ללא פרמטרים) כשהמשתמש נכנס לקישור auth/login, auth/logout, auth/register בהתאם.

בנוסף, הקובץ מכיל את הגדרת הקשטן (decerator) login\_required.

כשקוראים לפונקציה המקושתת בו, הוא בודק שיש משתמש במשתנה הגולבאלי (g) ואם לא, הוא מעביר את המשתמש לעמוד ההרשמה עם הודעה לפיה עליו להירשם לפני שהוא משתמש באתר. הוא מקשט את הפונקציות של העמודים הדורשים התחברות למערכת.

הקובץ מכיל גם את הפונקציה load\_logged\_in\_user שנקראת באופן אוטומטי כשמשתמש מגיע לעמודים register, login או logout הבודקת אם המשתמש שמור בסשין ואם כן שומרת או במשתנה הגלובאלי.

הקובץ model.py:

מכיל את ה blueprint model

הפונקציה view\_all טוענת את העמוד בו המשתמש רואה את כל המודלים שהועלו לאתר על כה.

הפונקציה get\_model\_id מחזירה את המזהה של מודל מתוך תטבלת המודלים בבסיס הנתונים בהינתן השם שלו. אם המודל לא נמצא בבסיס הנתונים – הפונקציה מוסיפה אותו.

הקובץ complition.py מכיל את ה blueprint complition.

הפונקציה index טוענת את העמוד הראשי בו מוצגות כל ההשלמות הקיימות בבסיס הנתונים.

הפונקציה create טוענת את העמוד בו המתמש מעלה כותב טקסט ובוחר מודל ופרמטרים ליצירת הטקסט. לאחר שהמשתמש לוחץ על הכפתור "Complete" המודל שהוא בחר משלים את הטקסט. ההשלמה נכנסת לבסיס הנתונים והמשתמש מועבר לעמוד הראשי.

השתמשתי ב Pytest בשביל לבדוק את כל האפליקציה. הפקודה pytest (ב command prompt מתוך התקייה final\_project/web\_app כאשר הסביבה הווירטואלית עובדת) מריצה את הקובץ conftest.py שמגדיר ומפעיל את כל הבדיקות.

מימוש הארכיטקטורה לטרנספורמר עם דיקודר בלבד:

ההסבר המלא על האלגוריתם של טרנספורמר עם דיקודר בלבד נמצא בסקירה על בינה מלאכותית.

בחלק זה אתמקד במימוש עצמו.

הקוד למימוש הארכיטקטורה בנספחים לקובץ זה.

המחלקה Transformer:

כל המודלים שאצור על מנת לאמן הם עצמים מהמחלקה Transformer שיורשת מ Model (שיורשת גם מ tf.keras,layers.Layer) ויש לה 3 פונקציות (בנוסף לאלו שהיא יורשת):

אתחול "\_\_init\_\_": שתחילה קוראת לבנאי של המחלקה Model, יוצרת קידוד מיקומים המתאים להייפר הפרמטרים של המודל ומשמש להגדרת ה decoder ואז יוצרת את השכבות: שיכון, דיקודר ו-(הכפלה במטריצת שיכון משוחלפת ולאחריה פונקציית softmax).

קריאה: "call": שמקבלת רשימה בה שני טנזורים: פלט (inp) הוא רצף הטוקנים שהמודל צריך להשלים שאורכו כאורך הטקסט המקסימלי ומטרה (tar) הוא רצף של טוקנים באורך השווה לאורך הפלט הרצוי. ובנוסף מקבלת משתנה training שנכון כשהמודל מתאמן ושגוי אחרת. למתודה זו קוראים באמצעות המתודה predict (קריאה ישירה ל call מביאה לשגיאה).

שאר המחלקות:

Decoder, DecoderBlock, PointWiseFeedForwardNetwork, MyMultiHeadAttention, ScaledDotProductAttention.

בנויות דומה למחלקה Tranformer: הן יורשות (באופן ישיר, בשונה מ \Tranformer) מ Layer ויש להן שתי פונקציות: אתחול וקריאה. פונקצית האתחול יוצרת את תתי השכבות ושומרות אותן במשתני מחלקה. בחלק מהמקרים היא שומרת גם הייפר-פרמטרים שהיא מקבלת, חישובי עזר ואת קידוד המיקומים. מתודת הקריאה מקבלת בנוסף למידע שמתואר בחלק התיאורטי גם מסכות שנוצרות בכל קריאה למודל.

הפונקציה positional\_encoding:

נקראת רק מתוך מתודת האתחול של Transformer. התוצאות שלה נשמרות במשתנה מחלקה במחלקות Encoder ו Decoder והן שימושיות במתודת הקריאה של המחלקות הללו.

הפונקציה create\_masks:

מקבלת את הרצף ואת הערך השלם שמייצג ריפוד של המודל ומחזירה מסכה אחת המכסה את המטרה ומסכה שניה המכסה את טוקן הריפוד. לפונקציה פונקציית עזר פנימית שנקראת create\_padding\_mask שמייצרת מסכה לכיסוי טוקן הריפוד שהיא גם חישוב עזר ליצירת מסכה למטרה. הפונקציה נקראית מתוך מתודת הקריאה של המחלקה SeTranformer.

הדמו:

הקובץ project\_demo.ipynb הוא מחברת ג'ופיטר אותה שאותה המשתמש פותח דרך colab .

במחברת המשתמש בוחר מודל מאומן מראש, ופרמטרים לדגימה (top k, top p, number of goups, group size) מכניס טקסט ורואה את הטקסט שהמודל חוזה בשיטת הדגימה שאני מציע בעבודה לעומת שיטות קודמות.

המודל המאומן מראש יכול להיות כל מודל שהועלו ל hugging face ויורש מהמחלקה [AutoModelForCausalLM](https://huggingface.co/docs/transformers/main/en/model_doc/auto#transformers.AutoModelForCausalLM). [בקישור הזה](https://huggingface.co/models?pipeline_tag=text-generation&sort=likes) אפשר למצוא רשימה מלאה של כל המודלים שמקיימים את הדרישות הללו.

הצגת התוצרים

בדיקת איכות הטקסטים שנוצרים על ידי האלגוריתם:

על מנת לבדוק את איכות הטקסטים שהאלגוריתם יוצר, בחרתי את סט הנתונים ted\_talks\_iwslt המכיל עשרות אלפי טקסטים שתורגמו על ידי מתרגמים מקצועיים.

נתנתי לאלגוריתם שלי לתרגם את הטקסטים ובדקתי את הקרבה בין התרגום שהאלגוריתם יצר לבין התרגום שהמתרגם המקצועי יצר בעזרת מדד ברט.

בניסויים שלי השתמשתי בדגימה מתוך הסתברות ללא מגבלות, במודל opt-125M ובטמפרטורה של אחת ושיניתי את גודל הקבוצה בלבד.

בחרתי בדגימה ללא מגבלות ובטמפרטורה של אחת על מנת לשמור על הערכי ברירת המחדל, במודל opt מכיוון שזהו מודל שפה סיבתי שאומן על תרגום טקסטים ארוכים ובגרסת 125 מיליון פרמטרים של המודל משיקולי חומרה (מגבלת זכרון של המעבד בו השתמשתי).

הקוד שבו השתמשתי על מנת להריץ את הניסויים נמצא בתיקייה evaluation.

את התוצאות אספתי ריכזתי [פה](https://www.comet.com/yonikremer/grouped-sampling-evaluation/reports) (שימו לב שהתוצאות מתעדכנות).

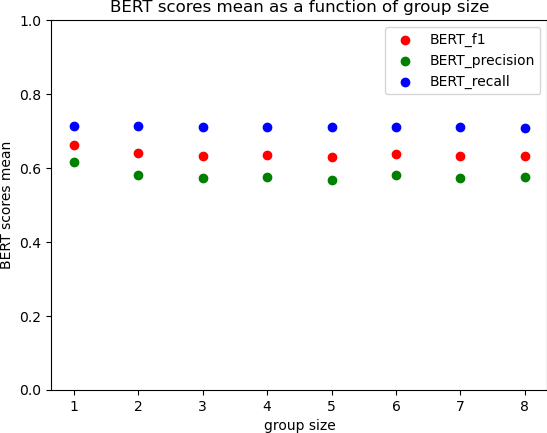
תוצאות – חלק ראשון

בניסויים הראשוניים הבנתי שבמשימת תרגום אני לא יודע מה יהיה אורך הפלט הרצוי אז שיניתי את הקוד של האלגוריתם ליצירת טקסט ככה שהטקסט יגמר כשהאלגוריתם יצור טוקן מיוחד שקיים בכל מודלי השפה שבא בסוף כל טקסט.

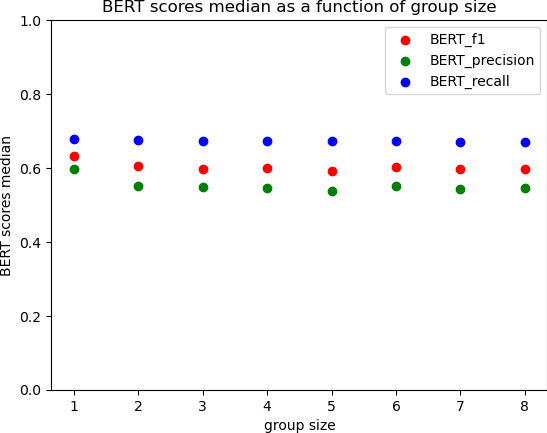
לאחר מכן שמתי לב שהרבה פעמים האלגוריתם יוצר טקסטים מאוד ארוכים (אלפי טוקנים) ולא מגיע לטוקן של סוף המשפט אז החלטתי שהאלגוריתם יעצור לאחר שהוא יוצר טקסט ארוך פי שתיים מהקלט. אורך זה בא מתוך סט הנתונים שלי – היחס המקסימלי בין האורכים של צמד טקסטים הוא כמעט 2.

אלו התוצאות שקיבלתי:

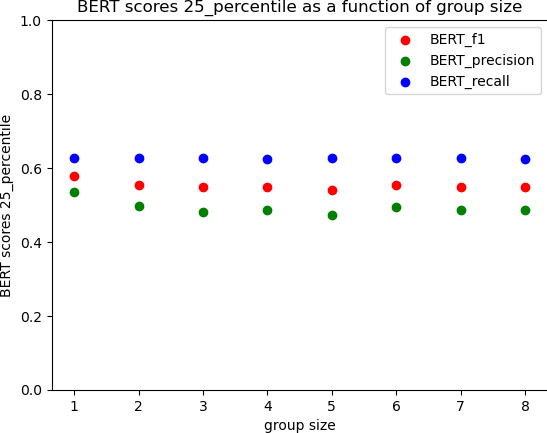
ממוצע:



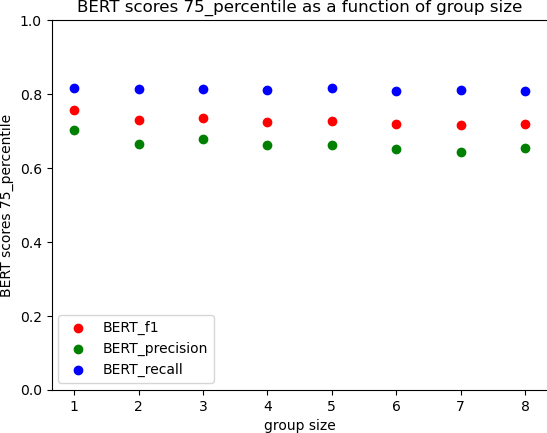
חציון:



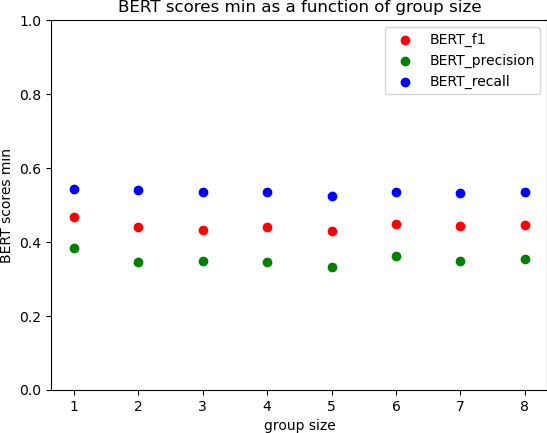
אחוזון 25:



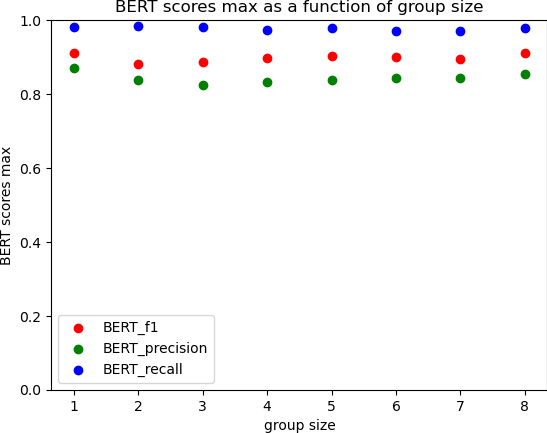
אחוזון 75:



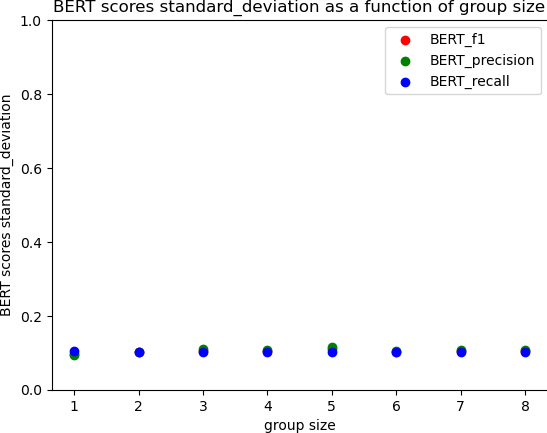
מינימום:



מקסימום:



סטיית תקן:



הדבר הראשון ששמתי לב אליו כמעט ואין השפעה על איכות הטקסטים. תוצאות אלו מראות על הצלחת המחקר בבירור – האלגוריתם משפר את זמן הריצה בלי לפגוע משמעותית באיכות התוצרים.

שמתי לב גם שה recall גדול משמעותית מה מה precision .

מתוך הנוסחאות (ראה סקירת בינה מלאכותית – מדד ברט):

כאשר x הוא המטריצה המתארת את טקסט המטרה ו y הוא הטקסט המועמד

ולכן הוא אורך הטקסט הרצוי ו הוא אורך הטקסט שהאלגוריתם יוצר.

ולכן אם

נחלק את הנוסחאות של המדדים ב (ידוע ש ) ונקבל

נכפיל ב (ידוע ששני האורכים חיוביים ולכן מכפלתם חיובית ואפשר להכפיל בה בלי לשנות סימנים) ונקבל:

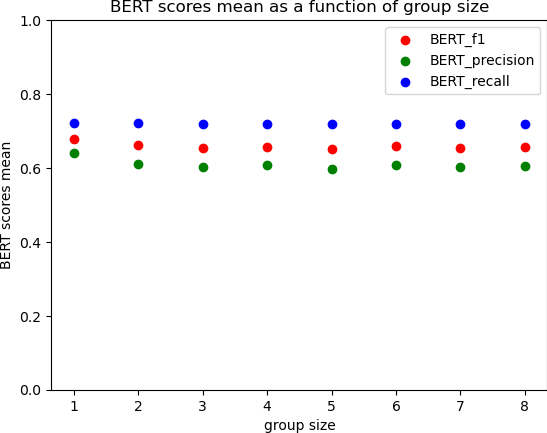
כלומר אורך הטקסט שהאלגוריתם יוצר גדול מאורך הטקסט הרצוי.

**תוצאות החלק השני**

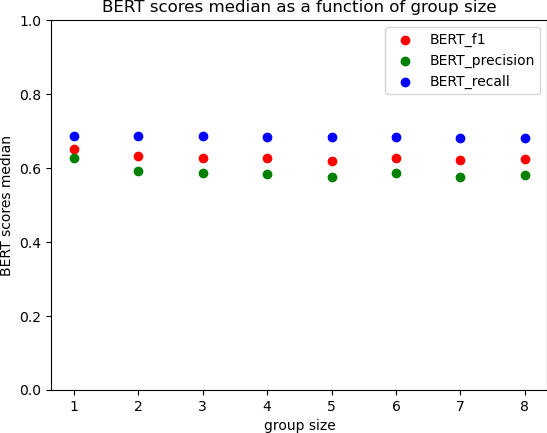
על מנת להקטין את אורכי הטקסטים שהאלגוריתם יוצר, שיניתי את הפרמטר answer\_length\_multiplier מ 2 ל 1.25 ככה שהאלגוריתם לא יצור טקסטים ארוכים ביותר מ 25% מהטקסטים שהוא מקבל.

מאחר ששיניתי פרמטר שעשוי להשפיע על איכות הטקסטים, התחלתי סדרת ניסויים חדשה עם הפרמטר החדש.

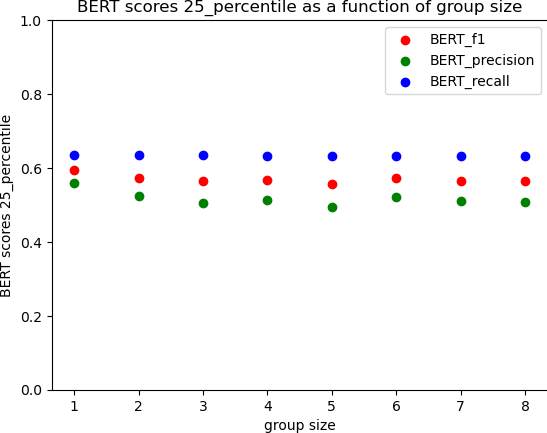
ממוצע:

:

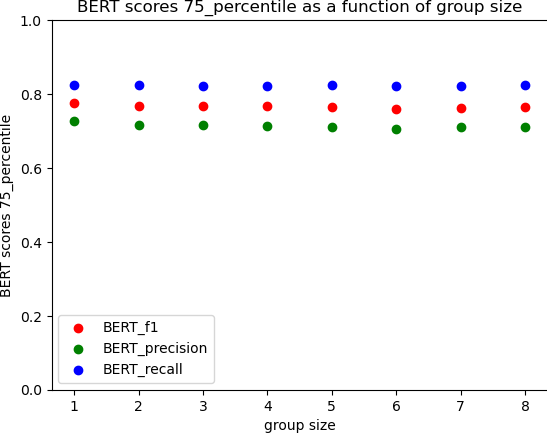
חציון:



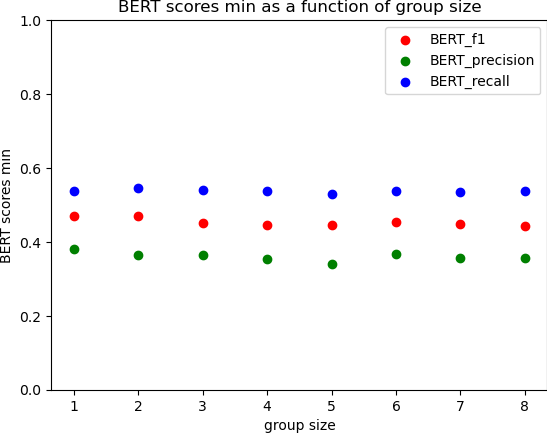
אחוזון 25:



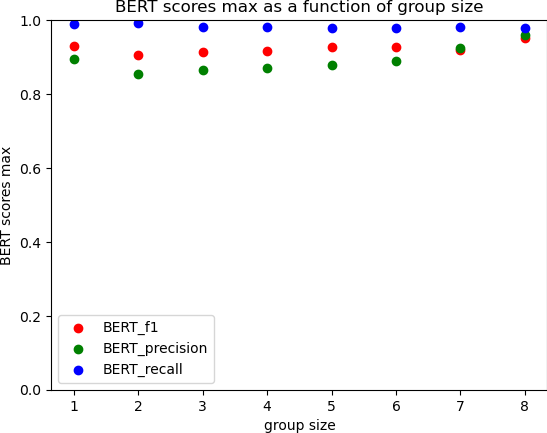
אחוזון 75:



מינימום:



מקסימום:



ראיתי שעדיין קיים פעם בין ה recall וה preccision אך הוא הצטמצם משמעותית והחלטתי שהמגבלה ששמתי לאורך הטקסטים היא מספיק משמעותית ואין צורך להקטין את הפרמטר answer\_length\_multiplier לפחות מ 1.25.

**חלק שלישי:**

בלי קשר לתוצאות הניסויים, שיניתי את מימוש האלגוריתם ככה ש:

במקום שההסתברות של טוקן שכבר מופיע בטקסט תקבע מלאכותית ל 0 – השתמשתי בשיטת "דגימה עם עונשים" (ראה סקירת בינה מלאכותית) עם θ=1.2 משום שזה הערך שהומלץ במאמר שהציג את השיטה.

בנוסף, מכיוון שהראיתי בחלקים הקודמים שהגדלת גודל הקבוצה מ אחד לגדלים קטנים יחסית שתיים עד שמונה) אינה פוגעת באיכות הטקסטים אני רוצה לבדוק מה קורה כאשר מגדילים את גודל הקבוצה מאוד. לכן בניתי ניסוי בו הגדלתי את גודל הקבוצה מאחד () ל שתיים () וככה הלאה עד חמש מאות ושתיים עשרה ().

ביבליוגרפיה

‘3. Data Model’. *Python Documentation*, Python Software Foundation, https://docs.python.org/3/reference/datamodel.html. Accessed 24 Dec. 2022.

Ackley, David H., et al. ‘A Learning Algorithm for Boltzmann Machines\*’. *Cognitive Science*, vol. 9, no. 1, Jan. 1985, pp. 147–69. *DOI.org (Crossref)*, https://doi.org/10.1207/s15516709cog0901\_7.

Agarap, Abien Fred. *Deep Learning Using Rectified Linear Units (ReLU)*. arXiv, 7 Feb. 2019. *arXiv.org*, https://doi.org/10.48550/arXiv.1803.08375.

Bridle, John. ‘Training Stochastic Model Recognition Algorithms as Networks Can Lead to Maximum Mutual Information Estimation of Parameters’. *Advances in Neural Information Processing Systems*, vol. 2, Morgan-Kaufmann, 1989. *Neural Information Processing Systems*, https://proceedings.neurips.cc/paper/1989/hash/0336dcbab05b9d5ad24f4333c7658a0e-Abstract.html.

‘Collections.Abc — Abstract Base Classes for Containers’. *Python Documentation*, The Python Software Foundation, https://docs.python.org/3/library/collections.abc.html. Accessed 24 Dec. 2022.

Eric V. Smith. ‘PEP 557 – Data Classes’. *Python Enhancement Proposals (PEPs)*, https://peps.python.org/pep-0557/. Accessed 24 Dec. 2022.

Fan, Angela, et al. ‘Hierarchical Neural Story Generation’. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, 2018, pp. 889–98. *DOI.org (Crossref)*, https://doi.org/10.18653/v1/P18-1082.

Guido van Rossum and Talin. ‘PEP 3119 – Introducing Abstract Base Classes’. *Python Enhancement Proposals (PEPs)*, https://peps.python.org/pep-3119/. Accessed 24 Dec. 2022.

‘Heapq — Heap Queue Algorithm’. *Python Documentation*, The Python Software Foundation, https://docs.python.org/3/library/heapq.html. Accessed 24 Dec. 2022.

Holtzman, Ari, et al. *The Curious Case of Neural Text Degeneration*. arXiv, 14 Feb. 2020. *arXiv.org*, http://arxiv.org/abs/1904.09751.

---. *The Curious Case of Neural Text Degeneration*. arXiv, 14 Feb. 2020. *arXiv.org*, https://doi.org/10.48550/arXiv.1904.09751.

Keskar, Nitish Shirish, et al. *CTRL: A Conditional Transformer Language Model for Controllable Generation*. arXiv, 20 Sept. 2019. *arXiv.org*, https://doi.org/10.48550/arXiv.1909.05858.

Kevin D. Smith, et al. ‘PEP 318 – Decorators for Functions and Methods’. *Python Enhancement Proposals (PEPs)*, 2 Sept. 2004, https://peps.python.org/pep-0318/.

Liu, Peter J., et al. *Generating Wikipedia by Summarizing Long Sequences*. arXiv, 30 Jan. 2018. *arXiv.org*, https://doi.org/10.48550/arXiv.1801.10198.

Mikolov, Tomas, et al. *Efficient Estimation of Word Representations in Vector Space*. arXiv, 6 Sept. 2013. *arXiv.org*, https://doi.org/10.48550/arXiv.1301.3781.

*Papers with Code - GPT Explained*. https://paperswithcode.com/method/gpt. Accessed 23 Dec. 2022.

*---*. https://paperswithcode.com/method/gpt. Accessed 23 Dec. 2022.

Press, Ofir, and Lior Wolf. *Using the Output Embedding to Improve Language Models*. arXiv, 21 Feb. 2017. *arXiv.org*, https://doi.org/10.48550/arXiv.1608.05859.

Rossum, Guido van. ‘Unifying Types and Classes in Python 2.2’. *Python*, https://www.python.org/download/releases/2.2.3/descrintro/. Accessed 24 Dec. 2022.

Suits, Daniel B. ‘Use of Dummy Variables in Regression Equations’. *Journal of the American Statistical Association*, vol. 52, no. 280, Dec. 1957, pp. 548–51. *DOI.org (Crossref)*, https://doi.org/10.1080/01621459.1957.10501412.

van Rossum, Guido, et al. ‘PEP 484 – Type Hints’. *Python Enhancement Proposals  (PEPs)*, https://peps.python.org/pep-0484/. Accessed 24 Dec. 2022.

Vaswani, Ashish, et al. *Attention Is All You Need*. arXiv, 5 Dec. 2017. *arXiv.org*, https://doi.org/10.48550/arXiv.1706.03762.

Warsaw, Barry, et al. ‘PEP 435 – Adding an Enum Type to the Python Standard Library’. *Python Enhancement Proposals (PEPs)*, https://peps.python.org/pep-0435/. Accessed 24 Dec. 2022.

Zhang, Tianyi, et al. *BERTScore: Evaluating Text Generation with BERT*. arXiv, 24 Feb. 2020. *arXiv.org*, https://doi.org/10.48550/arXiv.1904.09675.

.

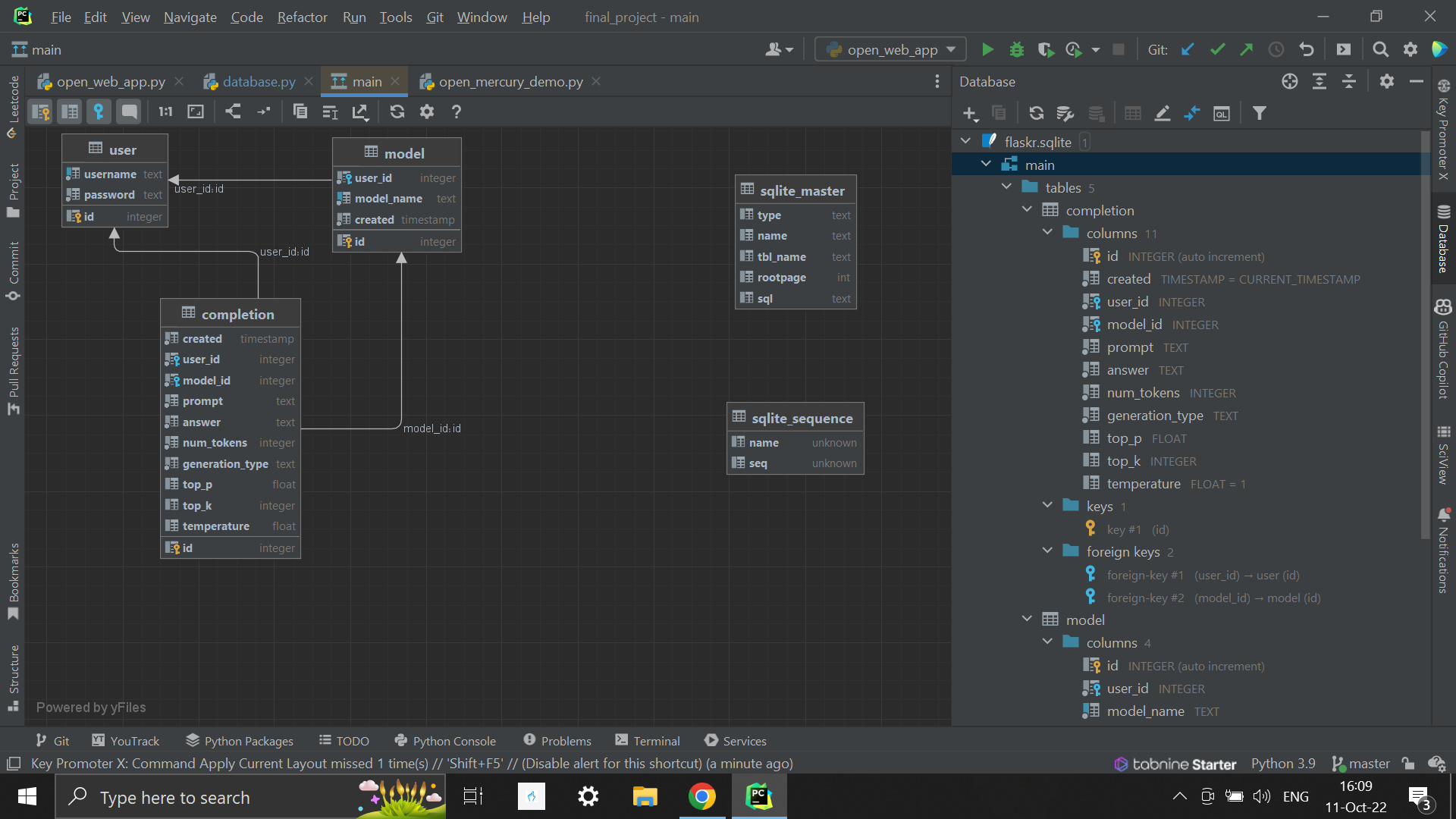
.

נספחים

חלקים נבחרים מהקוד:

דיאגרמת בסיס הנתונים של אפליקציית הווב:

בדיאגרמה הזאת ניתן לראות את פרטי בסיס הנתונים של אפליקציית הווב.



קישורים:

[העמוד של הפרויקט בגיטהאב](https://github.com/yonikremer/final_project)

[הדמו בקולאב](https://colab.research.google.com/github/yonikremer/final_project/blob/master/project_demo.ipynb)

דוקומנטציות:

[פייתון](https://docs.python.org/3/)

[פייתורץ'](https://pytorch.org/docs/stable/index.html)

[טנזורפלואוו](https://www.tensorflow.org/api_docs/python/tf/all_symbols)

[hugging face](https://huggingface.co/docs/transformers/main/en/model_doc/auto)

קורסים של אוניברסיטת סטנפורד:

[עיבוד שפה טבעית בעזרת למידה עמוקה](https://web.stanford.edu/class/cs224n/)

[הבנת שפה טבעית](https://web.stanford.edu/class/cs224u/)

מדריכים:

[NLP guide by PhD Lena Voita](https://lena-voita.github.io/nlp_course.html)

[data augmentation methods for text-blog](https://neptune.ai/blog/data-augmentation-nlp)

[Evaluating text generation blog post](https://towardsdatascience.com/how-to-evaluate-text-generation-models-metrics-for-automatic-evaluation-of-nlp-models-e1c251b04ec1)