**בינה מלאכותית**

**טוקניזציה**

מודלי שפה אינם מיצגים טקסט כרצף אותיות (מחרוזת) אלא כרצף של מילים, חלקי מילים או צירוף אותיות בעל משמעות (למשל הסיומת ים בעברית).

כשיוצרים טוקנייזר, 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.

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

כל מודל שקולט ו\או מייצר שפה מאומן בהינתן טוקנייזר – הטוקנייזר מוגדר לפני תחילת אימון המודל ולא משתנה אף פעם.

**אימון ראשוני לניתוח שפה בעזרת יצירת שפה**

**Generative Pre-Training (GPT) for Natural Language Proccessing**

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).

כאשר:

i המזהה של הטוקן שנמצא במקום ה i.

היא ההסתברות של הטוקן במקום ה t+1 להיות הטוקן שהמזהה שלו i.

seq len הוא אורך הרצף שהמודל מקבל.

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 (**Self Supervised Learning**):

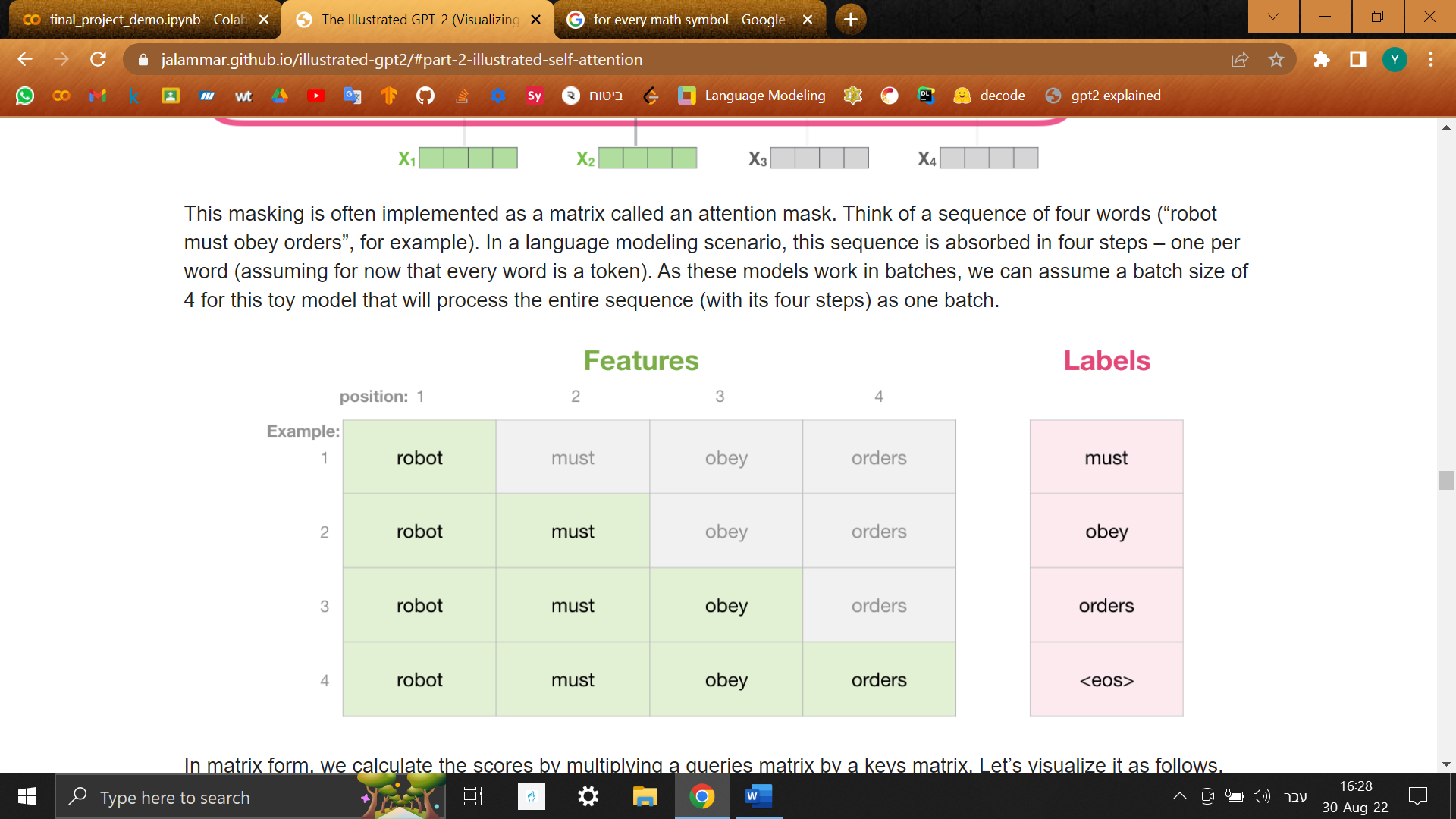
הנתונים עצמם אינם מחולקים לצמדים של קלט ופלט והמודל לומד לחזות או ליצור חלק אחד מהדוגמה בהינן חלק אחד מהדוגמה לכל דוגמה בסט הנתונים.

זאת בניגוד לשיטות למידה פופולריות אחרות:

לימוד מונחה (**Supervised Learning**) בו לכל דוגמה בסט הנתונים יש צמד קלט ופלט. לדוגמה מודל שמסווג תמונות שיש בהן מכוניות לתמונות שאין בהן מכוניות.

לימוד לא מונחה (**Unsupervised Learning**) בו אין חלוקה לקלט ופלט בכלל. לדוגמה: אם בסט הנתונים כל דוגמה היא ווקטור אז אפשר לאמן מודל שמוצא קבוצות של ווקטורים קרובים ומשייך אותם לאותה מחלקה.

חלקוה של המשפט "robots must obey orders" :



**למידה רב שלבית – Transfer Learning:**

המשימה שלחיזוי הטוקן הבא בטקסט אינה 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.
4. מתוך התפלגות של הטוקנים הפלטים בסט הנתונים.

**One Hot representetion:**

יצוג one hot ממיר אינדקס של מחלקה (במקרה שלנו טוקן) שהוא מספר שלם ואי שלילי (אפס או חיובי) לווקטור בינארי שאורכו מספר המחלקות בנתונים (במקרה שלנו מספר הטוקנים שהטוקנייזר שומר במילון).

יצוג one hot הוא בעצם ווקטור של הסתברות של קטגוריה ידועה.

היצוג נוצר לפי הפסודו-קוד:

one\_hot(class\_id, numer\_of\_cllasses):

ans = [0 for \_ in range(numer\_of\_cllasses)]

ans[class\_id] = 1

return ans

דוגמה:

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:

**טרנספורמר עם דיקודר בלבד – Decoder Only Transformer:**

במודל טרנספורמר מבוסס דיקודר בלבד בלוק הדיקודר הראשון מקבל את האמבדינג והקידוד המקומי של הקלט. שאר בלוקי הדיקודר מקבלים את הפלט של בלוק הדיקודר לפניהם.

הפלט של בלוק הדיקודר האחרון מוכפל במטריצת אמבידנג משוחלפת ליצירת מטריצת לוג'יט.

על מטריצת הלוג'יט מופעלת

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.

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

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

בעבודה שלי 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). ומטריצת לוג'יט לתאר מטריצה הבנויה מווקטורי לוג'יט כמו שמטריצת הסתברות בנויה מווקטורי הסתברות.

**בלוק דיקודר Decoder Block:**

בלוקי דיקודר הם שכבות עם תתי השכבות הבאות:

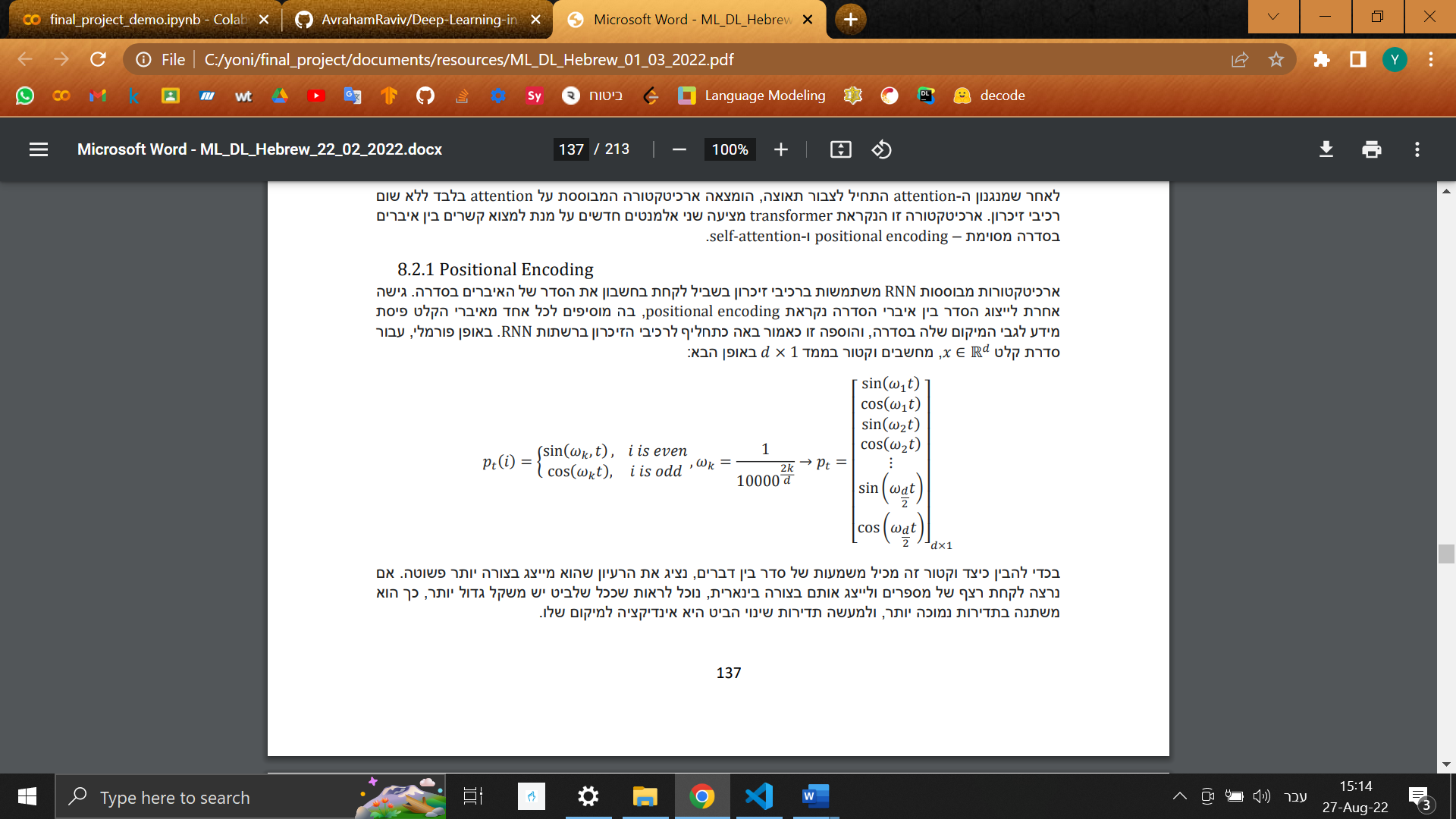
1. צומת לב עצמית רב ראשית עם מסכת הסתכלות קדימה
2. חיבור ונורמליזציה
3. רשת מחוברת לגמרי
4. חיבור ונורמליזציה

**קידוד מיקומי – Positional Encoding:**

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

, מחשבים וקטור במימד x 1 באופן הבא:



**אמבדינג – Embedding:**

אמבדינג 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 האמבדינג היא מכפלה של מטריצת הרצפים ביצוג one hot במטריצת האמבדינג.

קיימות מגוון שיטות ליצירת מטריצת אמבדינג שאינן בעזרת למידה עמוקה.

***הכפלה במטריצת אמבדינג משוחלפת:***

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

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

אינטואיציה:

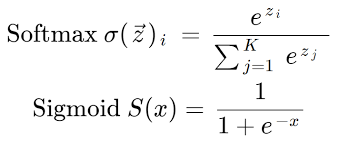
הדיקודר מחזיר את מטריצת אמבדינג של המילים במקום השני עד המקום seq len + 1 (לפני נורמליזציה).

הכפלה במטריצת אמבדינג היא תרגום של הסתברות לאמבדינג.

הכפלה במטריצת אמבדינג משוחלפת היא תרגות של אמבדינג להסתברות.

**הפונקציה softmax:**

*softmax() מוגדרת לפי הנוסחה:*

**

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

*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:***

*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.*

***מסכת הסתקלות קדימה look ahead mask***

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

*כאשר seq\_len הוא אורך הרצף.*

*המסכה נוצרת בצורה הבאה:*

*def create\_look\_ahead\_mask(seq\_len):*

*answer = zeros(shape=[seq\_len, seq\_len])*

*for i in range(seq\_len):*

*for j in range(seq\_len):*

*if j > i:*

*answer[i, j] = 1*

*או לפי ההגדרה 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**

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 dim).

נגדיר:

seq len = אורך הקלט לטרנספורמר (מספר חיובי ושלם)

*= המימד החבוי – תכונה של השכבה.*

*הן 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:*

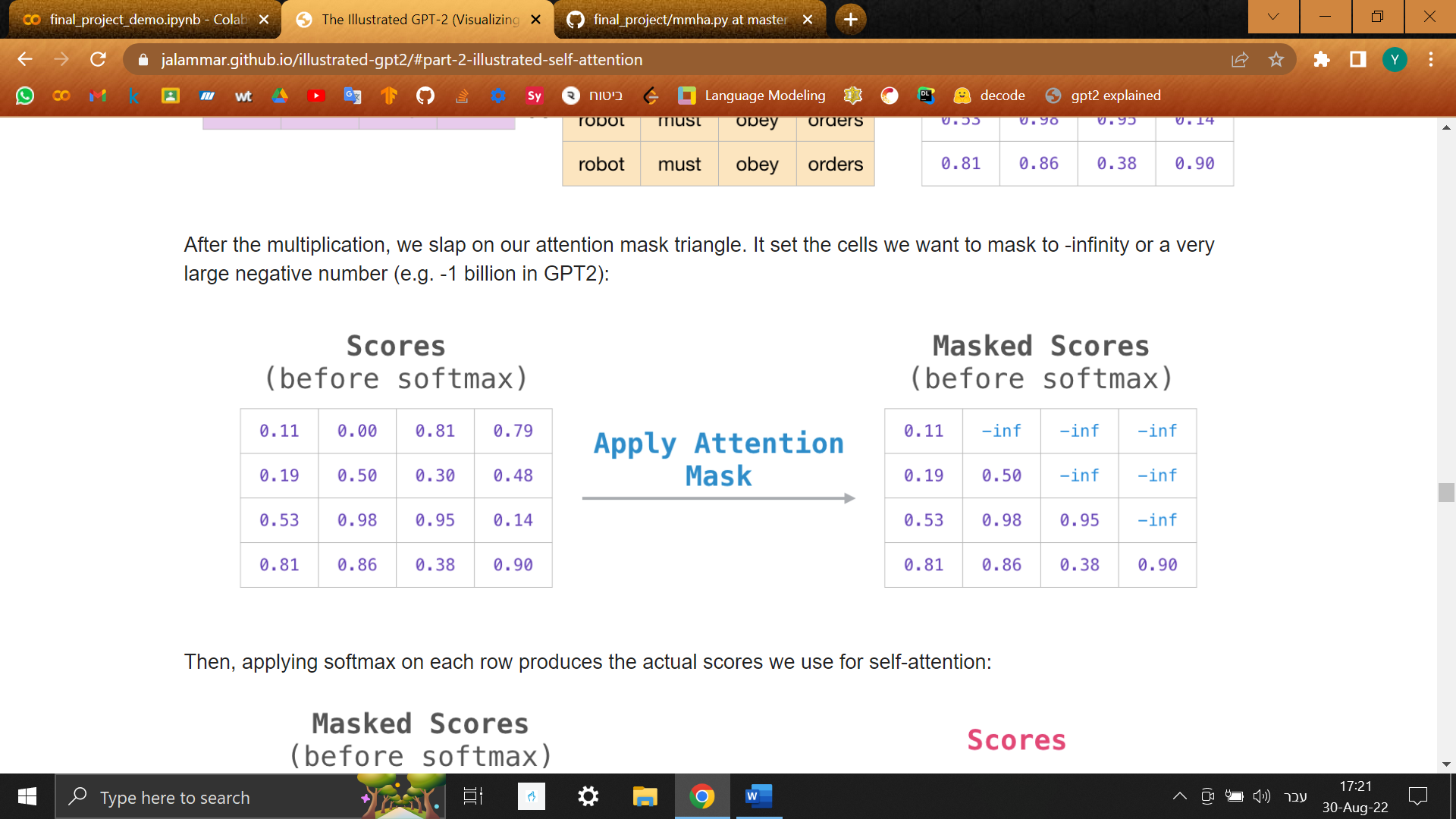
*את תוצאת המכפלה מחסרים ממטריצת הדמיון ככה שהדמיון בין*

*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.*



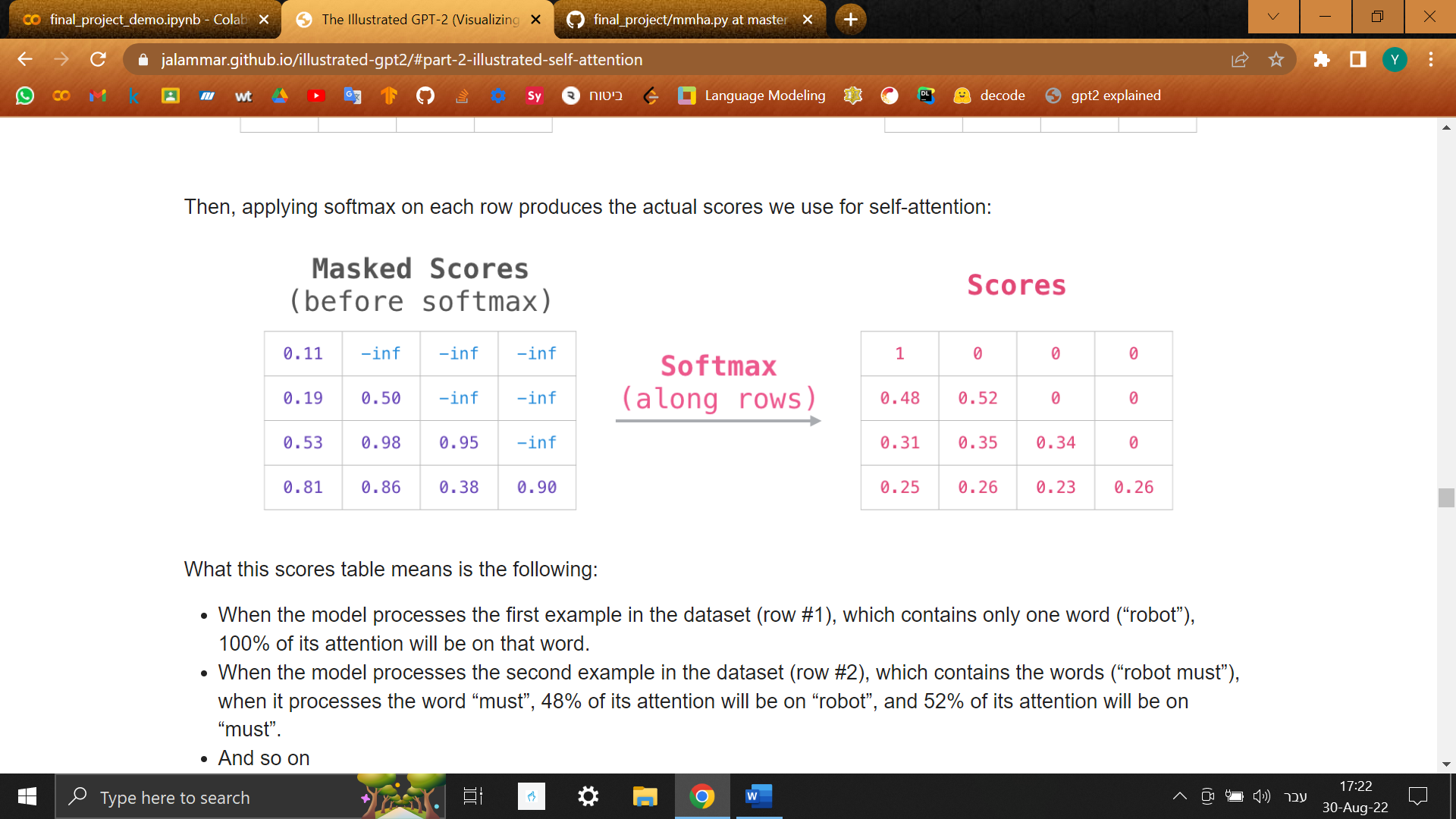
*הפעולה :*

*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:***

*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.*

***מסכת ריפוד - Pad/Padding Mask:***

*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.*

***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:***

*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:***

*הפונקציה פועלת על כל איבר בטנזור ונוחסתה:*

*הנגזרת של הפונקציה היא:*

***ריפוד - Padding:***

*בלמידה עמוקה אנחנו הרבה פעמי רוצים לשלוח למודל כמה דוגמאות בו זמנית.*

*עד עכשיו, אמרנו שהטרנספורמר מקבל רצף באורך שונה כל פעם אך קיימת שיטה לשלוח לטרנספורמר כמה דוגמאות בו זמנית וזה על ידי ריפוד.*

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

*לכל רצף ניצור מסכת ריפוד – ווקטור בינארי המוגדר לפי:*

***בלבול – Preplexity:***

*בלבול היא שיטה להעריך כמה המודל בטוח בתחזיות שלו כשהוא יוצר רצף.*

*הגדרה פורמלית:*

*כאשר הוא אורך הרצף שנחזה, היא ההסתברות של הטוקן במקום i (בהינתן הטוקנים הקודמים אליו).*

***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 אורך מסוים בו אנחנו רוצים שהטקסט המיוצר יהיה.*

*דגימה זה התהליך של רצף עצמו. ישנן שיטות שונות לדגימה.*

1. ***דגימה לפי הסתברות מקסימלית – argmax/greedy sampling:***

*נתחילה 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.*

*החסרונות של השיטה:*

* *המטרה של רוב הטקסטים אינה להיות כמה שיותר צפוים אלא להעביר מידע.*

*אם כל מילה תיהיה המילה הצפויה ביותר בהינתן המילים שלפניה אז כמות המידע שעובר תיהיה קטנה מאוד.*

* *שיטה זו נוטה ליצר את הטוקנים שמופיעים הרבה בסט האימון של המודל (בדרך כלל מילות קישור).*
* *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.*

1. ***דגימה רנדומלית מתוך הסתברות:***

*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 של השיטה:*

* *דגימה מגוונת.*

*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.*

1. ***דגימה מתוך k הטוקנים שהסתברותם הכי גבוה – Top k Sampling:***

*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*

*חסרונות של השיטה:*

* *לא מונעת לגמרי את האפשרות לדגום טוקנים שהסתברותם נמוכה.*
* *התעלמות מההתפלגות של ההסתברויות.*

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

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

*נבחר מספר p בין אפס ואחד.*

*אם p קטן או שווה להסתברות של הטוקן שהסתברותו הגדולה ביותר נבחר בו (בטוקן שהסתברותו הגודלה ביותר).*

*אחרת*

*ניצור ווקטור הסתברות חדש בו כל ההסתברויות 0.*

*נעבור על ווקטור ההסתברויות המקורי לפי סדר*

*נוסיף לווקטור החדש טוקנים כל עוד סכום ההסתברות לא עולה על p.*

*נחלק את ההסתברות של כל טוקן בסכום של הווקטור החדש (על מנת לקבל ווקטור שסכומו 1).*

*ונדגום מהווקטור שנוצר.*

*חסרון:*

* *אין מניעה לבחור הסתברות נמוכה.*

1. ***חיפוש עץ – Beam Search***

*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 כשמייצריםרצפים 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.*

1. ***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.*

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

***טמפרטורה - Temprature:***

*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.*