

# מודל שיחתי

מבוסס ארכיטקטורת טרנספורמר

מאת

אריאל צירינסקי



עבודת גמר במגמת בינה מלאכותית

חלופת למידת מכונה

בהנחייתה של לנה לינשיץ

מאי 2024

## תוכן עניינים

1.....	<b>מבוא</b>
1.....	הרקע לפרויקט ומטרתו
2.....	המצב הקיים בשוק
2.....	תהליך המחקר
2.....	אתגרים מרכזיים
3.....	הפתרון המוצע
8.....	<b>מבנה הפרויקט</b>
8.....	איסוף הכנה וניתוח הנתונים
8.....	מבנה מאגרי המידע
12.....	עיבוד הנתונים
15.....	ניתוח הנתונים הגומלים
23.....	שלב בניית ואימון המודל
23.....	אבני היסוד של רשת נוירונים ולמידה עמוקה
28.....	סוגי שכבות
44.....	תיאור UML של המחלקות
45.....	סמפלר
45.....	המודלים שעליהם בוצע האימון
49.....	פונקציית המחיר
51.....	אלגוריתם ההתכנסות (optimizer)
53.....	תהליך ותוצאות האימונים
61.....	שלב היישום
61.....	הפשטת היישום
61.....	שרת אינטרנט
62.....	שרת המחשוב
63.....	מנהל בקשות
65.....	תיאור UML של מחלקות היישום
66.....	<b>מדריך למפתח</b>

66.....	מבנה תיקיית הפרויקט
68.....	תוכן עניינים של המדריך למפתח
71.....	הכנת הנתונים, בניית ואימון המודל הראשון
97.....	הכנת הנתונים, בניית ואימון המודל שני
101.....	יישום
102.....	שרת אינטרנט
122.....	מנהל בקשות
128.....	שרת מחשוב
<b>149.....</b>	<b>מדריך למשתמש</b>
149.....	דרך סביבת פיתוח Google Colab
154.....	הרצת היישום המלא
154.....	הורדה והרצת הפרויקט
156.....	מבנה האתר
<b>161.....</b>	<b>רפלקציה</b>
<b>163.....</b>	<b>ביבליוגרפיה</b>
<b>167.....</b>	<b>נספחים</b>
167.....	מילון מושגים
168.....	דוגמאות לפלטי המודלים
168.....	המודל הראשון
177.....	המודל השני

## מבוא

### הרקע לפרויקט ומטרתו

מאז המצאת המחשב אנשים דמיינו לעצמם עתיד בו המחשבים יהיו חכמים יותר מבני האדם. מדעני המחשבים הגו תיאוריות למבנה המוח החשמלי, ויצרו תחום מחקר שלם לכך - מדעי הבינה המלאכותית. הם הרכיבו מבחנים שיחליטו האם אותות חשמליים חכמים מהאדם, ושערו מתי מהפכה זו תתחולל. סופרי המדע הבדיוני שכבר עשורים כתבו על המכונית המעופפת, ניסו לתאר עולם שבו המכונה יודעת יותר. עולם שבו לכל אדם עוזר אישי כל יודע וכל יכול.

הסדר הכרונולוגי התהפך וקיבלנו את המוח המלאכותי לפני המכונית המעופפת. בשנים האחרונות חלה התקדמות חסרת תקדים בטכנולוגיות עיבוד השפה הטבעית (NLP). מה שהתחיל כמודל שיכול לכתוב טקסטים חצי קוהרנטיים, הפך תוך פחות מחמש שנים למכונות ענק שיודעות לפענח שפות, תמונות, סרטונים וקוד טוב יותר מאנשים מן השורה, ולעיתים גם מומחים. המודלים השיחתיים בפרט, הפכו לכלי רב עוצמה, ממש כמו העוזרים האישיים הבדיוניים שתוארו לפני קרוב למאה שנה.

פרדיגמת הלמידה וחיפוש המידע השתנתה לחלוטין. לראשונה מזה כמה עשורים, אנשים עוזבים את מנועי החיפוש לטובת המודלים השיחתיים. כבר לא צריך לחפש בין אתרים מסורבלים עם מגילות של טקסט קשה להבנה. אפשר לשאול את העוזר האישי הכל יודע, שתוך כמה שניות ייתן מענה מפורט וברור עם קישורים חיצוניים למקורות המידע. ובמידת הצורך, העוזר האישי יכול להסביר נקודות לא מובנות, להשתמש במשלים ומטאפורות, לשנות את סגנון הדיבור והלמידה לפי העדפה אישית, ואפילו לשנות את שפת הדיבור כרצון המשתמש.

בעיית אי המודעות של החברה פוקדת את תחום הקיימות. גרעין הבעיה טמון ביכולת הנגישות לחומרי הלימוד. כדי להבין את סוגיות התחום צריך לקרוא עשרות מאמרים הכתובים בשפה שמברית העולם לא יודע או מתקשה להבין. נעשו מאמצים רבים ללמד את הציבור על חשיבות המחזור, חיסכון בחשמל, שמירה על ניקיון וכדומה. אולם, הקמפיינים מציגים מידע כללי מדי, ואילו כאשר מנסים לחקור את הדברים לעומק, פוגשים שוב את מגילות הטקסט המפחידות.

פרויקט זה מציע פתרון לבעיית חוסר הנגישות למידע בצורת מודל שיחתי. מודל זה יהיה בקיא בספרות המדעית של תחום הקיימות ומדיניות הסוגיה בישראל. אנשים יוכלו לדבר עם המודל וללמוד על הפן המדעי של סוגיות הקיימות וכיצד ניתן למגר נזקים סביבתיים. כך כמויות המידע לא ירתיעו את הקהל הרחב מללמוד על קיימות וליישם אורח חיים ירוק יותר.

בחרתי בנושא הפרויקט מכיוון שמודלים שיחתיים נמצאים בחזית המדעית של הבינה המלאכותית, ורציתי ללמוד כיצד הטכנולוגיות הללו עובדות. בנוסף, הפרויקט יכול לתרום רבות לתחום הקיימות ככלי

חינוכי. לדעתי, השקעה קטנה יחסית מצד הממשלה במודל כזה יניב תוצאות טובות הרבה יותר מהקמפיינים היקרים שראינו עד כה. הפרויקט נועד לקבוע קווים מנחים למודל עתידי.

## המצב הקיים בשוק

כיום, בשלהי מהפכת המודלים השפתיים, ניתן למצוא כלים רבים המקלים על חיפוש מידע. חלק מהמודלים, כדוגמת ChatGPT, לא יכולים לגשת לאינטרנט ולכן הידע שלהם מוגבל בתאריך אימון המודל. מנגד, יש מספר מודלים, כמו Bing AI, שיכולים לחפש ולאתר מידע באינטרנט שעוזר ביכולת שלהם לתת מענה לשאלות. עם זאת, לא קיים, לפחות בשעת כתיבת ספר זה, מודל המכוון לשאלות הקשורות לקיימות.

## תהליך המחקר

תהליך המחקר מורכב מסקירת ספרות מעמיקה לרבות קראית מאמרים מקצועיים, הרצאות מוקלטות ותכנים אחרים. נושאי הסקירה נבחרו לפי אזכורים במאמרים, הופעתם בסדרות הרצאות וקריאת פורומים. תהליך הלמידה ערך מספר חודשים וכלל בניית פרויקטים קטנים כדי לוודא בקיאות בנושאים, לרכוש אינטואיציה ולהכיר את תהליך יישום הטכנולוגיות. הסקירה הגיעה לסיום כאשר הרגשתי שהפרויקטים נהיו מספיק משוכללים ומטעם מגבלות זמן.

תהליך המחקר נערך בסדר הבא. נושאים שנלמדו בבית הספר אינם מוזכרים ברשימה, וחשוב לציין כי חלק מהטכנולוגיות לא ראו שימוש בתוצר הסופי.

1. מודל bigram וtrigram
2. מיפוי מילים (word embedding)
3. כימות טקסט (tokenization)
4. מודל RNN
5. מודל LSTM
6. מנגנון התשומת לב
7. ארכיטקטורת טרנספורמר המקורית
8. ארכיטקטורת מודל שפתי בעל טרנספורמר מפענח בלבד
9. אימון מודל מבוסס טרנספורמר
10. תהליך fine-tuning של מודל מבוסס טרנספורמר לתת מענה לשאלות

## אתגרים מרכזיים

ישנם מספר אתגרים בהם נתקלתי במהלך העבודה על הפרויקט. רשימה זו כוללת את האתגרים המרכזיים שהקשו בתהליך הלמידה, איסוף הנתונים, אימון ויצירת היישום.

1. נגישות למידע - מודלים מבוססי טרנספורמר נראו לראשונה בשנת 2018, ומודלים שיחתיים מבוססי טרנספורמר שיישמו את הטכנולוגיה נהיו פופולריים בשנת 2022. על כן, מספר המקורות העוסקים בבניית ובאימון מודלים מבוססי טרנספורמר הינו מוגבל. מספר המקורות המתעדים את תהליך הפיכת מודלים לשיחתי קטן עוד יותר. לכן הייתי צריך להסתמך על מאמרים מקצועיים כמקור המידע העיקרי.
2. כוח מחשוב - מודלים שפתיים גדולים דורשים משאבי מחשוב אדירים. בין אם מדובר באימון פרויקטים קטנים, אימון המודלים הסופיים ותהליך ניסוי וטעיה הייתי צריך גישה למחשבים רבי עוצמה. לא יכולתי להסתמך על שירותי מחשוב ענן מכיוון שאלו היו גובים מאות דולרים על עשרות שעות האימון. החלטתי להשתמש במחשב הנייח שלי, בעל כרטיס מסך RTX 2060 עם 6GB של זיכרון, כדי לאמן את המודלים. אולם גרסאות הtensorflow החדשות אינן תומכות במערכת ההפעלה windows. חלק מאימוני המודלים ערכתי על גרסאות ישנות של tensorflow מטעמי נוחות. כדי לאמן את המודלים הסופיים השתמשתי בסביבת לינוקס.
3. מודלים שפתיים גדולים צורכים מאגרי מידע עצומים. מציאת מאגר שאלות תשובות בפרט היה מאתגר מאוד. לאחר מחקר רב מצאתי מאגר עם זוגות של שאלות תשובות במגוון נושאים. כדי למקד את המודל בנושא הקיימות, אספתי כמה אלפים של שאלות ותשובות בסוגיות הקיימות באמצעות מודלים יצרניים.

## הפתרון המוצע

המודל המוצע בפרויקט זה מנבא את התפלגות ההסתברויות לאסימון הבא בהינתן הקשר, משימת סיווג. בתהליך יצירת התוכן, המודל מקבל הקשר המכונה prompt ומנבא את האסימון הבא. מוסיפים את האסימון לprompt וחוזרים על הפעולה חלילה עד שמתקבל אסימון שמסמן את סוף הקטע. כעת הprompt מכיל את הקלט המקורי, בנוסף לפלט של המודל.

בתום המחקר המקדים החלטתי להשתמש במודל המבוסס על ארכיטקטורת טרנספורמר. למודלים מבוססי טרנספורמר יש מספר יתרונות בהשוואה למודלים שפתיים אחרים כדוגמת CNN ו LSTM. היתרון המובהק ביותר הוא גודל טווח ההקשר (context window) ואיכות החיפוש בו. מודלים שפתיים אחרים סובלים משכחה. ככל שמילים רחוקות יותר, יכולת המודל לזהות קשרים ביניהן קטנה, ולאחר מרחק מסוים הוא שוכח מילים כמעט לחלוטין. לעומת זאת, לטרנספורמר יש זיכרון מושלם שאינו תלוי

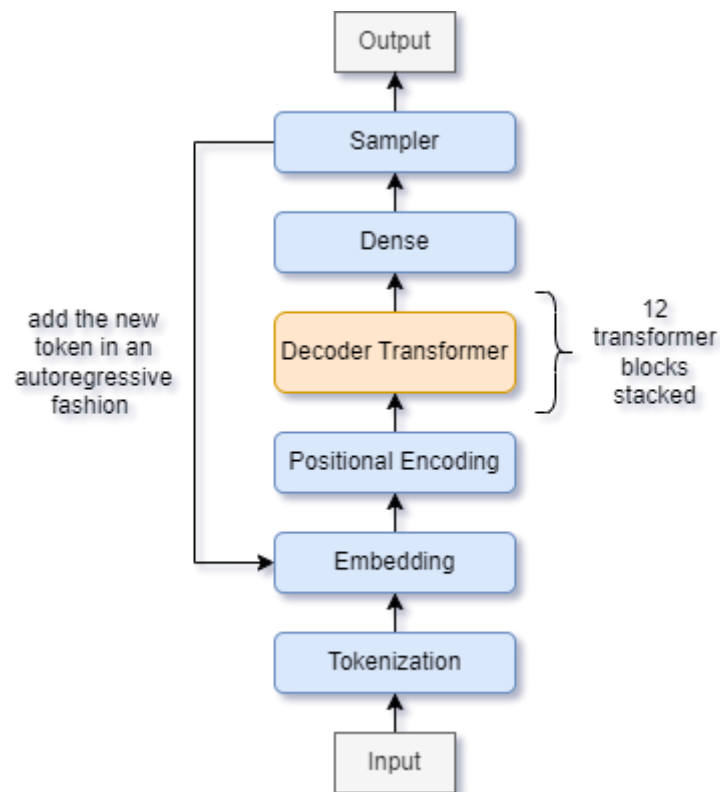
באורך הקלט, ותאורטית אין מגבלה לטווח ההקשר שלו<sup>1</sup>. יתרון נוסף הוא יעילות האימון. בעוד שמודלים אחרים צריכים לעבור חלחול קדימה עובר כל אימון במאגר הנתונים, ארכיטקטורת הטרנספורמר מאפשרת לאמן מספר אימונים במקביל בחלחול קדימה אחד.

ישנם מספר דרכי כימות של מילים ואף משפטים לוקטורים. בפרויקט זה מוצגת שיטה המורכבת משני שלבים עיקריים. בשלב הראשון הופכים את הקלט הטקסטואלי למספרים דיסקרטיים המכונים אסימונים (tokens). שלב זה מתבצע באמצעות האלגוריתם Byte Pair Encoding מכיון שהוא מסוגל לקודד כל קלט לאוצר מילים של אסימונים מוגבל, ולהתקנות לכל אימון משמעות לשונית גדולה. לאחר מכן, נעשה שימוש בשכבת הטמעה (embedding layer), שהופכת כל אימון לוקטור בעל גודל  $d_{model}$ . לבסוף, האסימונים עוברים בשכבת קידוד מיקום, מכיון שהמנגנונים של הטרנספורמר לא יודעים להסיק מידע מיקומי בכוחות עצמן.

כאשר יוצרים את התוכן יש חשיבות גדולה לשיטת בחירת האסימונים. המודל עצמו לא מנבא את האסימון הבא, אלא מחשב את התפלגות ההסתברויות לאסימונים השונים. לו היינו בוחרים כל פעם את האסימון בעל ההסתברות הגדולה ביותר, היינו מקבלים פלט לא יצרתי. חמור עוד יותר, למודלים שפתיים יצרנים יש נטייה לחזור על רצפי מילים ואף משפטים שלמים שוב ושוב. כדי לשפר את איכות הפלט של המודל נעשה שימוש באלגוריתם המכונה סמפּלר (sampler) שתפקידו לבחור אסימונים מתוך ההתפלגות. נבחר הסמפּלר  $top\_p$  אשר בוחר מתוך קבוצת אסימונים בעלי ההסתברות הגדולה ביותר שסכומם אינה עולה על ערך  $p$  מוגדר.

---

<sup>1</sup> תאורטית, אורך הקלט אינו מוגבל, אולם הזיכרון שיש להקצות למודל גדל בצורה ריבועית ביחס לטווח ההקשר. עבור קלטים ארוכים מאוד דרישות הזיכרון עולות על יכולות המחשוב הקיימות.



איור 1: דיאגרמה המתארת את החלקים השונים של המודל הסופי. הקלט עובר *tokenization*, הטמעה, קידוד מיקום, יחידות טרנספורמר, שכבת *dense* ולבסוף נבחר אסימון על ידי הסמפלר. אם האסימון אינו מסמל את סוף הפלט, מוסיפים אותו לקלט וחוזרים על התהליך שוב.

כדי לאמן מודל שיחתי נדרש מאגר מידע ענקי, מגוון ואיכותי. אולם אין דרך פרקטית ליצור מאגר מידע של שאלות ותשובות בגודל המבוקש. לכן, מחלקים את אימון המודל לשני שלבים. בשלב הראשון המודל לומד את השפה, כלומר הוא לומד לנבא את האסימון הבא בהינתן הקשר מסוים. מעבר למבנים תחביריים וקשרים בין מילים, בשלב הראשון המודל גם לומד את הידע שלו על העולם. רכישת השפה ולמידת עובדות צורכת כמות אדירה של מידע מגוון. בשלב השני המודל לומד את תבנית המענה על שאלה. מאחר והרשת כבר יודעת את השפה ועובדות על העולם, אפשר להסתפק במאגר מידע מאוד קטן ביחס למאגר המידע הקודם.

במסגרת הפרויקט אימנתי שני מודלים שונים, אחד לכל שלב המתואר לעיל. את המודל הראשון בניתי ואמתני מאפס, ללא שימוש במודלים קיימים. אולם, בחלק השני של האימון השתמשתי במודל קיים עם מאגר מידע שהכנתי. הסיבה לכך היא שבכדי לאמן מודל שפתי קוהרנטי, הוא צריך מאות מיליוני פרמטרים, ואף מיליארדי פרמטרים. אימון מודל בגודל כזה מאפס יעלה סכומי עתק במחשוב ענן ויזהם



את הסביבה לשווא<sup>2</sup>. ההבדל היחיד בארכיטקטורה בין המודל בחלק הראשון לבין המודל בחלק השני הוא מספר יחידות הטרנספורמר וגודל ההטמעה.

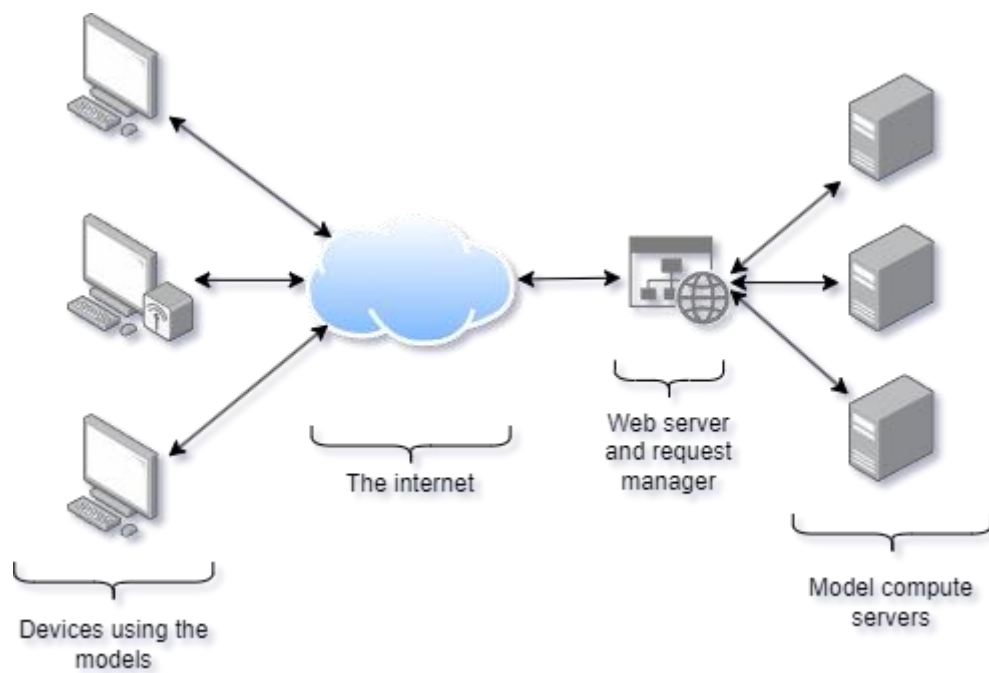
יישום המודל נעשה באמצעות אתר אינטרנט ומערכת שרתי מחשוב. האתר כולל הסבר קצר של הפרויקט וגישה לממשקים המאפשרים אינטראקציה עם שני המודלים הסופיים. עבור המודל השיחתי הממשק מדמה צ'אט שבו המשתמש יכול לשלוח הודעה המכילה שאלה, והמודל משיב בהודעה המכילה את התשובה. הממשק המודל הראשון הוא תיבת טקסט שבה המשתמש יכול לכתוב התחלה של כתבה, והמודל ימשיך לכתוב אותה.

השרת שמריץ את האתר אינו מתפעל את המודלים ואינו מריץ את חישובם. לשם כך נבנתה מערכת שרתי מחשוב המורכבת משרת ראשי המנתב את בקשות המחשוב ושרתי מחשוב שמריצים את המודלים. השרת הראשי רץ עם שרת האינטרנט באותה מכונה תחת אותו process. כאשר האתר מקבל בקשה לתקשר עם המודל, הוא פונה אל שרת המחשוב הראשי אשר מנתב את בקשתו לשרת מחשוב שמחזיר את תשובת המודל.

הבחירה ליצור מערכת שרתי מחשוב כחלק מהיישום התקבלה משתי סיבות עיקריות. ראשית, מחשב אחד אינו יכול לעמוד בדרישות המחשוב האדירות של מודלים שיחתיים. מאחר ומטרת הפרויקט כוללת את שמימוש המודל על ידי מספר רב של משתמשים, מערכת שרתי מחשוב היא חלק חיוני ביישום. שנית, חברות המציעות גישה למודלים שיחתיים עושות שימוש במערכות דומות. כדי להבין טוב יותר את עולם הבינה המלאכותית מעבר לתאוריה, החלטתי שיישום המערכת הוא חלק הכרחי.

---

<sup>2</sup> עלות היאמון של מודל בגודל 110 מיליון פרמטרים משוער ב\$3751 לבין \$12571. טביעת הרגל הפחמנית של האימון קרובה ל-1500 קילוגרם פחמן, השווה לטיסה בין יושבית.



איור 2: דיאגרמה המתארת את עבודת היישום. מכשירים המחוברים לאינטרנט מתקשרים עם האתר שמציג בפניהם ממשק לשימוש במודלים. כאשר משתמש מפעיל מודל, האתר מפנה את בקשות המחשוב לשרתי מחשוב המחוברים עליו, ומעביר את התוצאות למשתמש.

## מבנה הפרויקט

### איסוף הכנה וניתוח הנתונים

#### מבנה מאגרי המידע

כמתואר במבוא, אימון מודלים שיחתיים נעשה בשני שלבים עיקריים כדי לעמוד בדרישה לכמות מידע אדירה. המטרה של השלב הראשון היא ללמד את המודל את השפה ועובדות על העולם. בתום האימון הראשוני המודל צריך להיות בקיא במבנים תחביריים, ביטויים, הגדרות וקרשים בין מושגים. חלק זה דורש מאגר מידע גדול במיוחד ומגוון מאוד. אין חשיבות לאיכות המידע ועדיף כמות ומגוון מאשר איכות.

מרבית המודלים השיחתיים התאמנו על מאגר המידע OpenWebText או מאגרים דומים לו. פרויקט בניית מאגר הנתונים נעשה בכדי לדמות את המידע שעליו התאמן המודל GPT2 ולאפשר לקהל הרחב ליצור מודלים יצרנים ללא השקעה כספית מופרזת בהשגת מידע. כדי למצוא מקורות למאגר, החוקרים יצרו תוכנה שמחפשת קישורים באתר Reddit שנשלחו על ידי משתמשים אנושיים. לאחר מכן הם הורידו את כל התכנים מהקישורים, סננו מידע שאינו באנגלית ומחקו כפילויות. לבסוף הם נותרו עם מאגר מידע שגודלו 40GB המונה 8,013,769 מסמכים.

עם זאת, מאגר הנתונים OpenWebText גדול מדי להיקף פרויקט זה. שימוש במאגר המידע מצריך קידוד עבור tokenizer הספציפי למודל שנבנה במסגרת הפרויקט. תהליך זה היה לוקח מספר שבועות. בנוסף, כזכור המודל הראשון לא יעבור את האימון השני שהופך אותו למודל שיחתי. לכן, החלטתי להשתמש במאגר מידע קטן יותר, אך גדול ומגוון מספיק כדי לאפשר למודל ללמוד את השפה ועובדות בסיסיות על העולם.

בחיפוש אחר במאגרי מידע העמודים בדרישות הפרויקט, מצאתי מאגר מידע בשם cnn\_dailymail. מאגר זה מכיל 311,971 כתבות מהעיתון CNN שפורסמו ב-DailyMail. במקור המאגר נועד ללמידת רשת המסכמת כתבות, ולכן לכל כתבה מצורף גם סיכום קצר. הסיכומים מופשטים מדי וכתובים בטון שונה מהכתבות עצמן. על כן, הם לא שימשו בתהליך למידת המודל.

'id': d7783bd2bf5ad92156962380342411c2eb068853

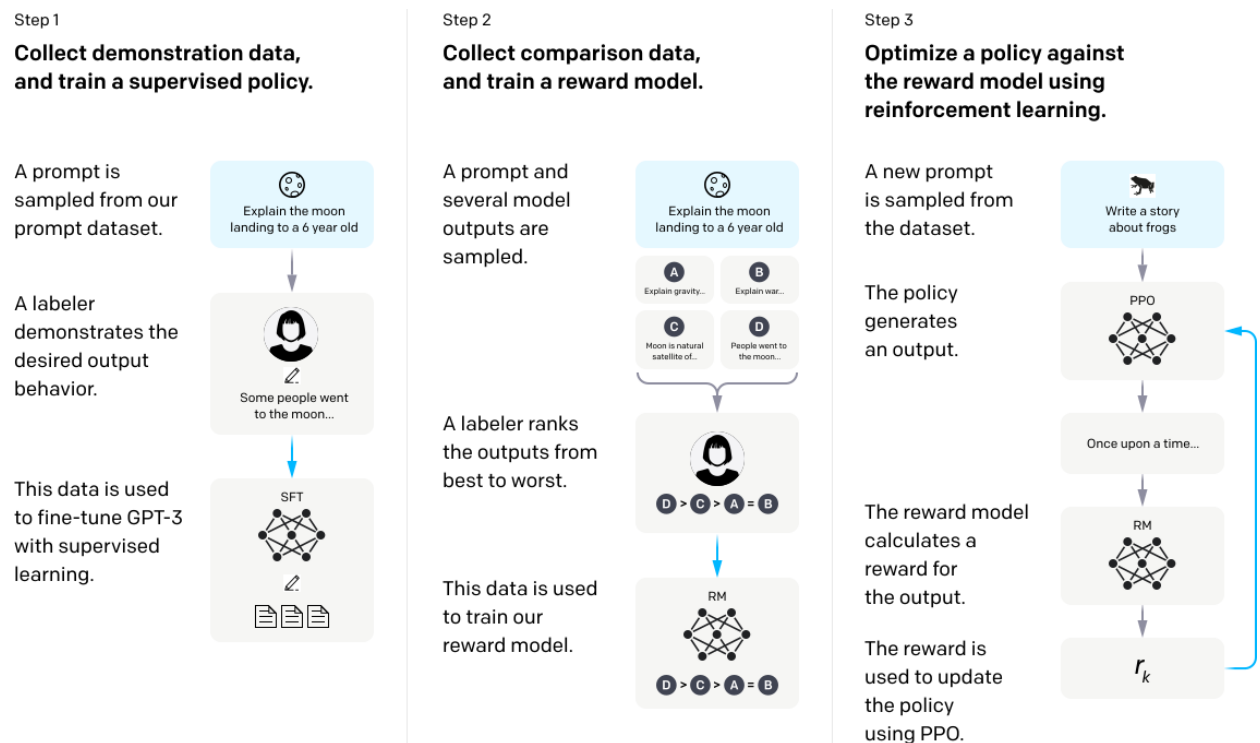
'article': '(CNN) -- An American woman died aboard a cruise ship that docked at Rio de Janeiro on Tuesday, the same ship on which 86 passengers previously fell ill, according to the state-run Brazilian news agency, Agencia Brasil. The American tourist died aboard the MS Veendam, owned by cruise operator Holland America. Federal Police told Agencia Brasil that forensic doctors were investigating her death. The ship's doctors told police that the woman was elderly and suffered from diabetes and hypertension, according to the agency. The other passengers came down with diarrhea prior to her death during an earlier part of the trip, the ship's doctors said. The Veendam left New York 36 days ago for a South America tour.'

'highlights': 'The elderly woman suffered from diabetes and hypertension, ship's doctors say. \nPreviously, 86 passengers had fallen ill on the ship, Agencia Brasil says.'

טקסט 1: דוגמה לפריט במאגר הנתונים cnn\_dailymail. ה"id" מכיל hash של הכתבה, "article" מכיל את התכבה עצמה ו"highlights" מכיל סיכום של הכתבה.

האימון השני נועד ללמד את המודל את מבנה השאלה-תשובה, הרי לאחר האימון הראשון המודל יודע רק להמשיך וליצור תוכן. במידה והיו שואלים אותו שאלה, הוא עלול היה לחשוב שזו כותרת והיה כתוב מאמר, או לחשוב שיוצרים רשימת שאלות ואף לענות עם עוד שאלות ועוד כל מני מקרים. לכן, מאגר המידע השני צריך להכיל זוגות של שאלות ותשובות בדגש על איכותן. לעומת גדלי העתק של המאגר הראשון, ניתן להסתפק במאגר יחסית קטן בשלב זה, מכיוון שהמודל כבר יודע את השפה ומכיר עובדות על העולם.

כדי לבנות מאגרי מידע עבור מודלים שיחתיים גדולים, כדוגמת ChatGPT, נוצרו ונאספו זוגות של שאלות ותשובות ידניות. חברות אלה שכרו עובדים אשר כתבו שאלות ולאחר מכן תשובות בסגנון הדיבור של המודל. כאשר סיימו ליצור את מאגר המידע, הם אימנו מודלים יצרניים פשוטים, כאלה שעברו רק את השלב הראשון של האימון הדו-שלבי. בנוסף, חברות כמו OpenAI אימנו מודל שני שיכול לדרג תשובות של המודל לפי איכותן. המודל המדרג הצליח לשפר את המודל השיחתי עוד יותר, כך שיצור פלטים הדומים יותר לפלטים בעלי דירוג גבוהה.



איור 3: תהליך הפיכת GPT3 למודל שיחתי ממודל יצרני רגיל

עם זאת, בהיקף פרויקט זה אין מימון כדי ליצור מאגר מידע כמתואר לעיל. בנוסף, מאגרי המידע של חברות הבינה המלאכותית נחשבים לנכסים חשוב מאוד, ולכן לא פורסמו. לאחר מחקר מעמיק מצאתי פרויקט של אוניברסיטת סטנפורד שבו ערכו את האימון השני על מודל יצרני בשם Alpaca של חברת מטה. במאמר שנכתב על הפרויקט, הצוות הסביר כיצד יצר מאגר מידע מספק בעלות של \$500 בלבד, ואף פרסם את המאגר בשלמותו.

כדי ליצור את מאגר השאלות-תשובות הצוות השתמש במודל בינה מלאכותית קיים, בפרט במודל היצרני Text-davinci-003 של חברת OpenAI. הם הסבירו למודל שברצונם ליצור מאגר שאלות ותשובות כדי ללמד מודל שפה גדול, הוסיפו רשימה של הנחיות וצירפו 175 דוגמאות שכבתו ידנית. המודל עיבד את המשימה וכתב 52,000 זוגות של שאלות-תשובות במגוון תחומים. מאגר זה ישמש כבסיס למאגר של האימון השני ומטרתו ללמד את המודל לענות על שאלות באופן כללי.

כמות השאלות הקשורות לסוגיות הקיימות הייתה בלתי מספקת. לכן, החלטתי להשתמש בשיטה דומה לזו של פרויקט סטנפורד וליצור מאגר מידע נוסף הדוגל בשאלות קיימות. יצרתי קלט מתאים עם רקע למטרת המשימה, הנחיות למשימה ודוגמאות לשאלות. לאחר מכן, השתמשתי במודל ChatGPT 3.5 כדי ליצור את המידע. אחרי מספר שעות של עבודה השגתי 1500 שאלות-תשובות. סיננתי את השאלות כדי לשמור על איכות המידע ונותרתי עם 1077 זוגות.

You are asked to come up with a set of 20 question-answer pairs on the topic of sustainability. These questions will be given to a GPT model in order to fine tune it for answering sustainability questions. The questions should be in the theme of someone trying to understand what causes climate change and what they can do to prevent it.

Here are a few examples of questions following the theme:

- 1 .What causes global warming?
- 2 .Should I be worried about climate change?
- 3 .What can I do to prevent climate change?
- 4 .What are some tips to reduce my carbon footprint?
- 5 .Are electric cars better for the environment?
- 6 .How can I start recycling?
- 7 .What will happen if we don't stop global warming?
- 8 .How can I get to work without my car?
- 9 .How much CO2 does a plane emit?
- 10 .How can I educate others about climate change?

Here are the requirements:

- 1 .The language used for the questions should be diverse.
- 2 .The questions should touch on a wide range of topics in climate change.
- 3 .The questions should be in English.
- 4 .The questions should be 1 to 2 sentences long, and in the theme of the example questions.
- 5 .The answers should be an appropriate response to the questions. Make sure the answers are less than 100 words.

List of 20 question-answer pairs:

טקסט 2: קלט שהוזן למודל בינה מלאכותית יצרני כדי ליצור מאגר מידע של שאלות-תשובות בנושא קיימות.

בנוסף, נוצר מאגר מידע שלישי שמטרתו ללמד את המודל אישיות. המידע נאסף באותו אופן כמו המאגר השני, אולם נאספו רק 100 זוגות של שאלות-תשובות. בין הפריטים ניתן למצוא שאלות אודות שמו של המודל, מדוע נוצר, כיצד נוצר ומי יצר אותו. לבסוף, כל שלושת מאגרי המידע הולחנו יחד כדי ליצור את מאגר המידע הסופי לאימון השני, לפני תהליך הכימות.

'question': 'What is the main driver of global warming?'

'answer': "The primary driver of global warming is the increased concentration of greenhouse gases, such as carbon dioxide, in the Earth's atmosphere."

טקסט 3: דוגמה לזוג שלא-תושבה שנוצר על ידי בינה מלאכותית יצרנית כחלק ממאגר המידע השני.

## עיבוד הנתונים

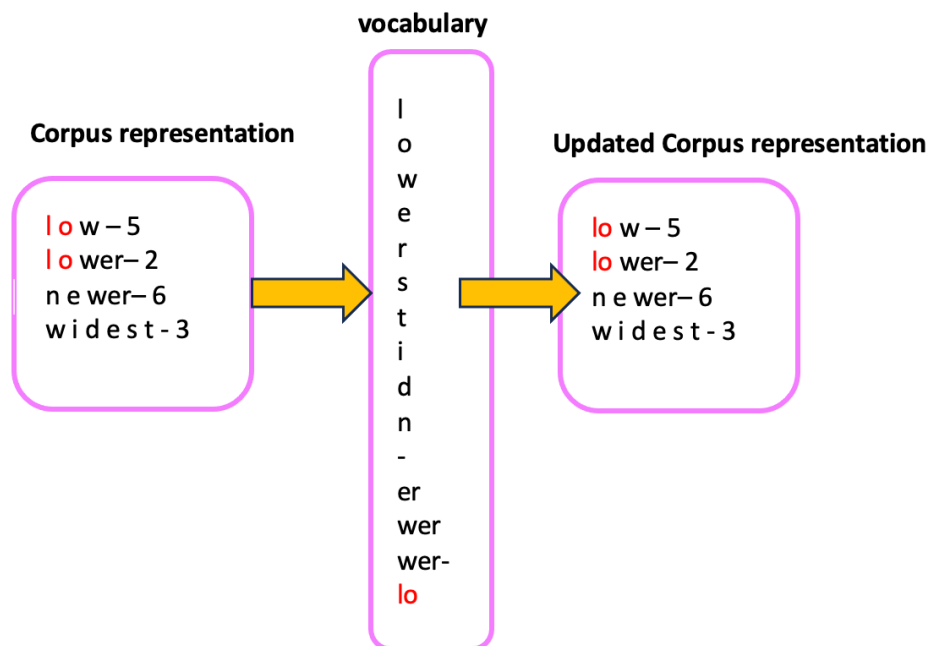
מחשבים שומרים ומייצגים טקסט כערכים בינאריים דיסקרטיים. כל תו מיוצג על ידי אחד או שניים בייטים, תלוי בסוג הקידוד. מודל הבינה המלאכותית לא יכול לפענח ערכי `ascii`, לכן יש להמיר אותם לערכים הנמצאים באוצר המילים שלו, ערכים אלה מכונים אסימונים. המודל אינו מכיר במילים, אלא רק באסימונים. לכן, גם כאשר הוא יוצר טסטים הוא יוצר אותם אסימון אחר אסימון. תהליך הפיכת טקסט לאסימונים מכונה `tokenization`, כמו כן תהליך המרת אסימונים לטקסט מכונה `detokenization`. לפני שאפשר לעבוד עם מאגר נתונים טקסטואלי יש קודם כל להמירו לאסימונים.

שיטת הקידוד הפשוטה ביותר היא הפיכת כל תו לאסימון, אולם יש בכך כמה בעיות. החיסרון המרכזי הוא שכל אסימון מכיל משמעות לשונית קטנה. על המודל קודם לבנות מן התווים מילים שלמות, ורק לאחר מכן להסיק מסקנות. בנוסף, כל משפט מורכב ממספר אדיר של תווים ביחס למספר המילים. הדבר מוביל לא רק למאגר נתונים עצום, אלא גם להסקה פגומה של המודל. לבינה מלאכותית מבוססת טרנספורמר יש טווח הקשר קבוע מראש. אילו היו האסימונים מילים שלמות, טווח ההקשר היה יכול להכיל טקסט שלם, בעוד שאסימונים המהווים תווים יכולים להכיל כמה משפטים בטווח המוגבל.

שיטה נוספת היא למפות כל מילה בשפה לאסימון. שיטה זו אמנם פותרת את בעיית המשמעות הלשונית הקטנה של כל אסימון, אך היא גם כרוכה בבעיות. במילון אנגלית מודרני יש כ-170,000 מילים, אולם הוא אינו מכיל את כל המילים האפשריות באנגלית, ואינו מכיל שמות כלל. ויקיפדיה כוללת 4,270,000 הגדרות עבור השפה האנגלית. אך גם היא אינה מכילה את כל המילים האפשריות ואת כל השמות הקיימים. בנוסף, אוצר מילים כה גדול מוביל למודל בעל מספר פרמטרים אדיר. לו היו מתגברים על כל המכשולים הללו מצליחים לבנות מודל כזה, תוצאותיו היו מאכזבות. הרוב המוחלט של האסימונים יופיעו רק כמה פעמים במאגר הנתונים, והמודל לא ילמד לפענח וליצור אותם.

קיימים מספר אלגוריתמים אשר מצליחים לתת מענה לכל הבעיות. כמתואר במבוא, האלגוריתם `Byte Pair Encoding`, בקיצור `BPE`, נבחר במסגרת פרויקט זה. עבודת האלגוריתם פשוטה ויעילה מאוד. הוחלט לממש את המקודד מאפס, ולכן נוצרה מחלקה בשם `tokenizer` המלכדת את הפונקציונאליות שלו: אימון, קידוד טקסט ופענוח טקסט מקודד.

בשלב האימון אלגוריתם BPE מתחיל עם אוצר מילים בסיסי המכיל את כל התווים האפשריים. הוא מחלק את הטקסט למילים לפי רווח, שומר את השכיחות שלהם מטעמי יעילות, ומקודד אותם לפי אוצר המילים הבסיסי. כעת הוא מחפש בין המילים את זוג האסימונים השכיח ביותר, מלחים אותם לאסימון חדש ומעדכן את המילים לפי האסימון שהתווסף. האלגוריתם חוזר על פעולה זו עד שהוא מגיע לאוצר מילים בגודל שנקבע מראש. ישנם מספר אסימונים המכונים "אסימונים מיוחדים" (special tokens). האסימונים מייצגים מידע תבניתי על הטקסט, למשל תחילת טקסט, סוף טקסט ריפוד וכד'. בגללן משמעותן הייחודית, לא נעשה עליהן אימון והן לא נכללות באסימונים אחרים. בתהליך הקידוד, המודל מלחים אסימונים לפי טבלת ההלחמה שנוצרה בזמן האימון. כדי לפענח קטע מקודד, הוא הופך כל אסימון לייצוג הטקסטואלי שלו.

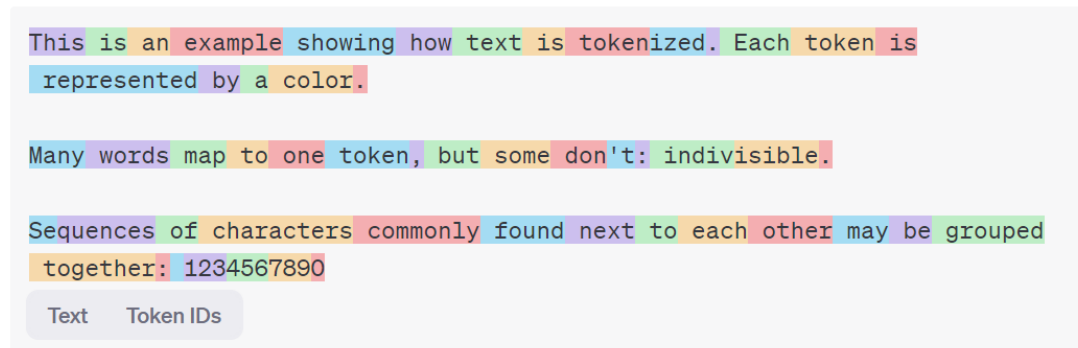


איור 4: תהליך קידוד מילים לאסימונים. tokenizer עובר על המילים בקורפוס ומצוא אסימונים שניתן להלחים כדי לקבל מילה שקיימת שאוצר המילים שלו. בדוגמה הזו, האלגוריתם מלחים את האסימונים l o-l ואסימון אחד lo.

שני המודלים משתמשים באלגוריתם BPE, אולם עם אוצר מילים שונה. המקודד של המודל הראשון מכיל 2500 אסימונים שכולם באותיות קטנות. לעומת זאת, המקודד של המודל השני מכיל 50,257 אסימונים עם אותיות גדולות. עבור המודל הראשון בחרתי ליישם את האלגוריתם בעצמי. תהליך אימון tokenizer וקידוד הטקסט לוקח זמן רב. דרך אחת לייעל אותו היא לנצל במספר ליבות במעבד במקום לרוץ על ליבה אחת. החלטתי לא לייעל את האלגוריתם מאחר והתהליך כרוך בבניה מחדש של פסי המידע וכתביה מחדש של חלק ניכר מהקוד. האלגוריתם אינו מנצל את כוח המחשוב של כרטיס המסך, ולכן הרצתי את האימון והקידוד על המחשב הנייח שלי. כל התהליך לקח קרוב ליומיים.



Tokens	Characters
54	241



איור 5: דוגמה לקטע טקסט המקודד באמצעות אלגוריתם BPE כאשר כל צבע מייצג אסימון. ניתן לראות כי מילים שכיחות ואף רצפי תווים כמו 123 מיוצגים על ידי אסימון יחיד. לעומת זאת, מילים נדירות כמו indivisible מחולקות למספר אסימונים.

בתהליך האימון המודל צריך לקבל זוג של רשימת אסימונים באורך טווח ההקשר, המוזהבים ערכי הX וY. הזוג מבוסס על אותו קטע טקסט, כאשר האיבר המכיל את הפלט של המודל מוזז אסימון אחד ימינה. סידור זה של הנתונים מאפשר למודל ללמוד לנבא את האסימון הבא בהינתן הקשר מסוים. תהליך יצירת בסיס הנתונים עבור כל מודל מעט שונה עקב האילוצים השונים של כל אחד וסגנון המידע השונה.

מאגר המידע של המודל הראשון, שכזכור מבוסס על כתבות, מחולק על בסיס כל כתבה כדי לא לערבב ביניהן. לאחר שמקודדים כתבה לאסימונים מוסיפים אסימון בתחילתה, המסמל את תחילת הכתבה, ואסימון בסופה המסמל את סוף הכתבה. כדי ליצור את זוגות הקלט ופלט לאימון, יש לחלק את רשימת האסימונים לחלקים שגודלם שווה לאורך טווח ההקשר פלוס אחד, במידת הצורך מוסיפים אסימוני ריפוד. בסיס הנתונים אינו מכיל את זוגות הקלט ופלט מטעמי מקום אחסון. לכן, כאשר טוענים את מאגר הנתונים ממירים כל איבר לזוג הקלט ופלט.

מאגר המידע עבור המודל השני מורכב מזוגות של שאלות ותשובות. את הזוגות מכניסים לתבנית שמכילה את השאלה ומסבירה למודל שעליו להשיב בהתאם. המטרה היא לגרות את החלקים המתאימים "במוח" של המודל, כלומר שיבין שהוא מתבקש לענות על שאלה. לאחר מכן, מקודדים את הפריטים ומוסיפים אסימוני ריפוד במידת הצורך. במודל זה אורך טווח ההקשר הינו 2048, אולם מטעמי יעילות הדוגמאות מקבלות אורך של 512 אסימונים. ישנם 3 דוגמאות העולות על אורך זה, והן הושמטו. למודל עצמו יש מנגנון שיכול לאמן את רשת באמצעות דוגמת טקסט אחת, שאינה מחולקת לקלט ופלט. על כן, אין צורך לחלק כל פריט במאגר הנתונים לפלט וקלט מראש.

Below is an instruction that describes a task. Write a response that appropriately completes the request.

### Instruction:

{instruction}

### Response:

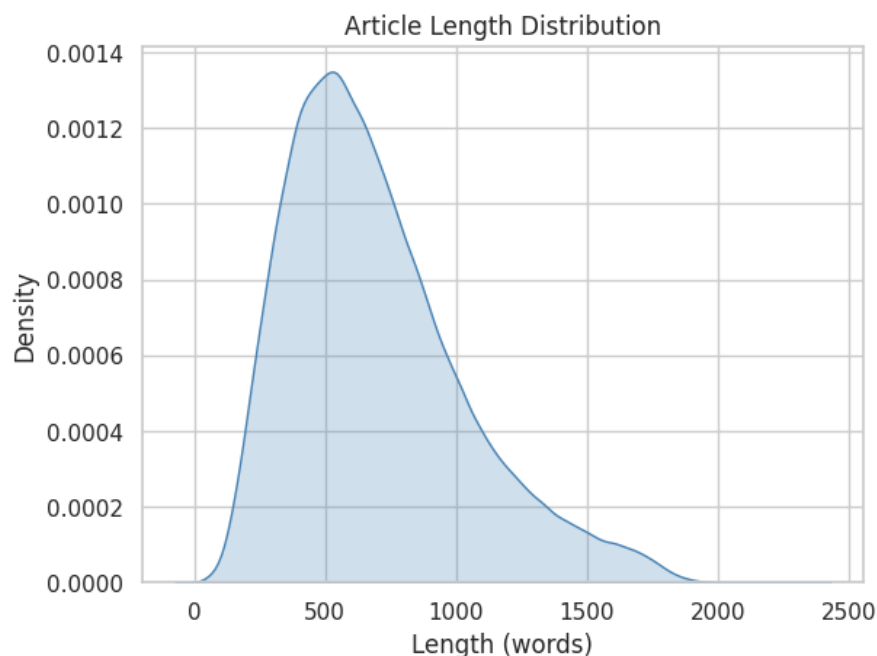
{response}

טקסט 4: תבנית השאלה-תשובה שהמודל מקבל בתור קלט.

תבנית השאלה מכילה מספר לא זניח של אסימונים שמשפיעים על ערך פונקציית המחיר. התבנית אינה משתנה בין שלאות, ולכן המודל לא לומד מניסיון לבא את התבנית. כדי לדייק את ערך פונקציית המחיר, ובכך לשפר את תהליך האימון, נעשה שימוש ב-sample weights. משקולות הפריטים הינו פרמטר בפונקציית fit של tensorflow, שקובע את משקל ההפסד של כל איבר על ערך ההפסד של המודל. אסימונים שמרכיבים את התבנית מקבלים משקל של 0.05, ואילו האסימונים שמרכיבים את התוכן עצמו מקבלים משקל של 1.

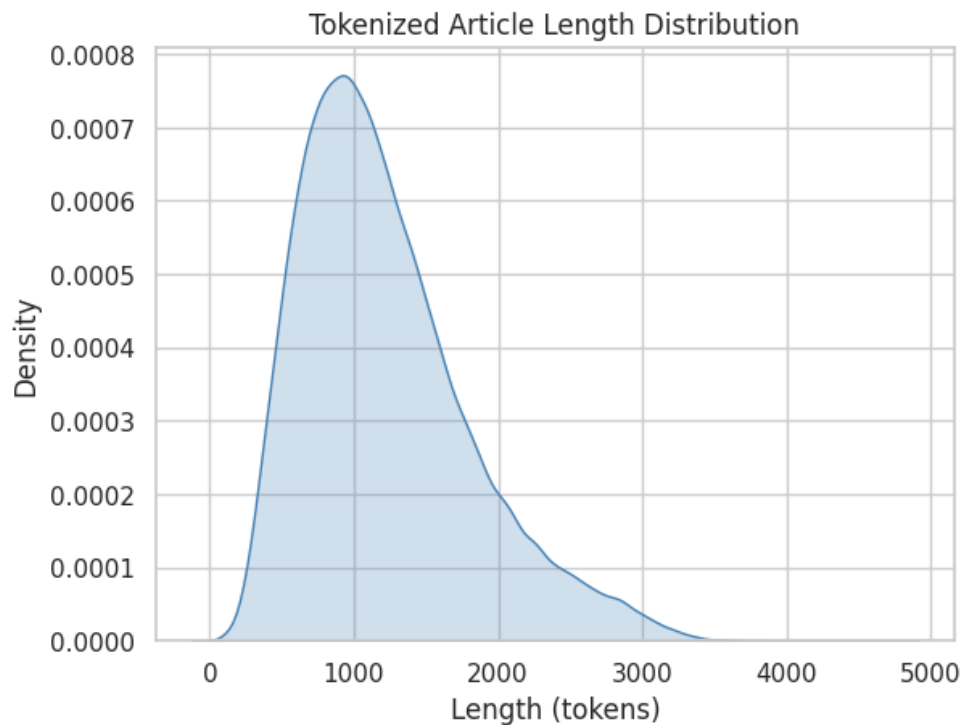
### ניתוח הנתונים הגומלים

מאגר המידע הראשון



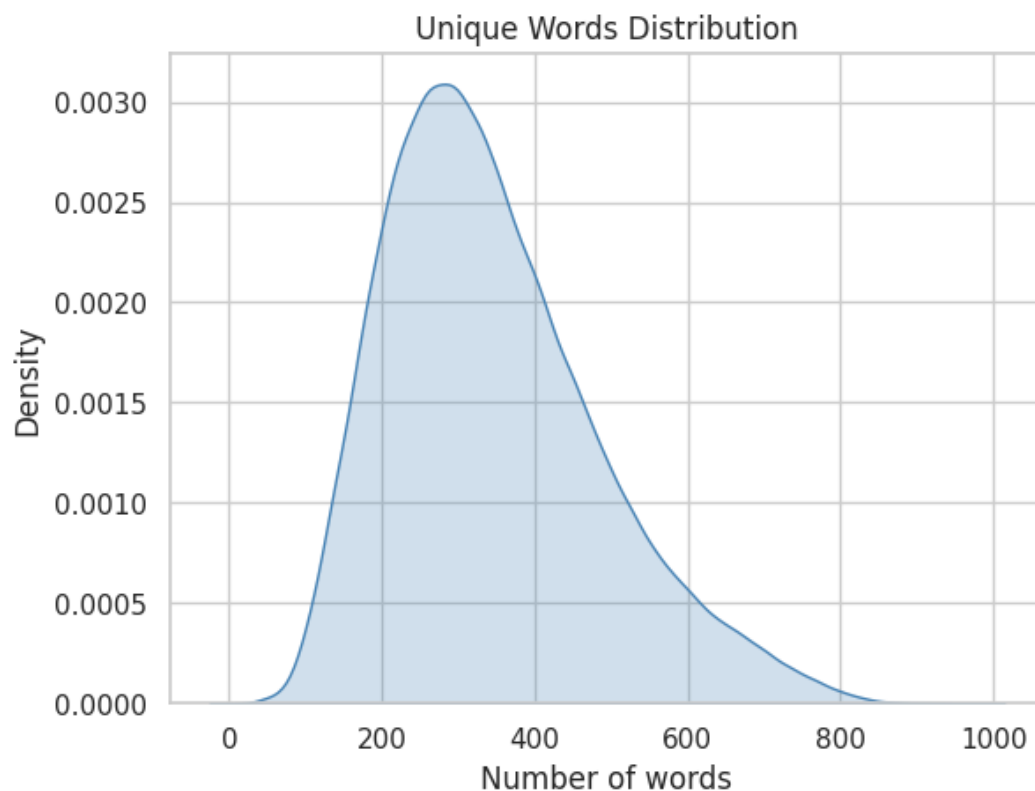
איור 6: התפלגות מספר המילים בכל כתבה.

כפי שהיה אפשר לשער, התפלגות אורכי הכתבות לפי מספר המילים מזכירה התפלגות גאוסית. האורך השכיח ביותר הוא בקירוב 550, והממוצע נמצאת בקירוב 750. מספר הכתבות בעלות יותר מ-2000 מילים הוא זניח.



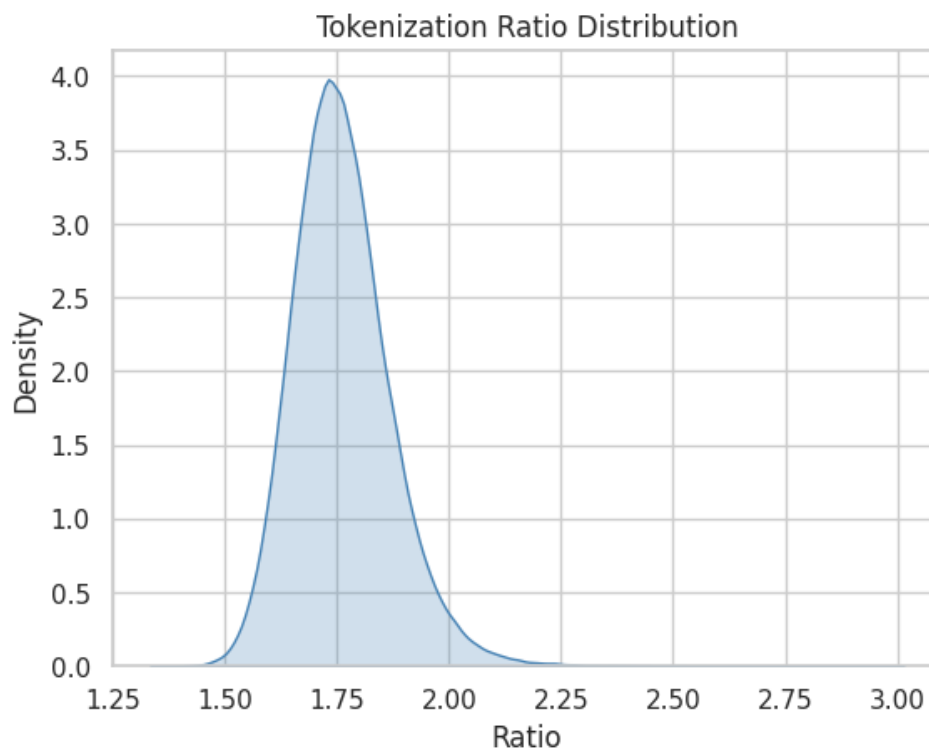
איור 7: התפלגות מספר האסימונים של כל כתבה אחרי *tokenization*.

בגלל אופי תהליך הקידוד של הכתבות לאסימונים, ההתפלגות אורכי הכתבות באסימונים דומה לזו של התפלגות אורכי הכתבות במילים, אם כי סטית התקן היחסית קטנה יותר. לכן, צורת הפעמון תלולה יותר. שכיח ההתפלגות הוא בקירוב 950 והממוצע 1200.



איור 8: מספר המילים הייחודיות בכל כתבה.

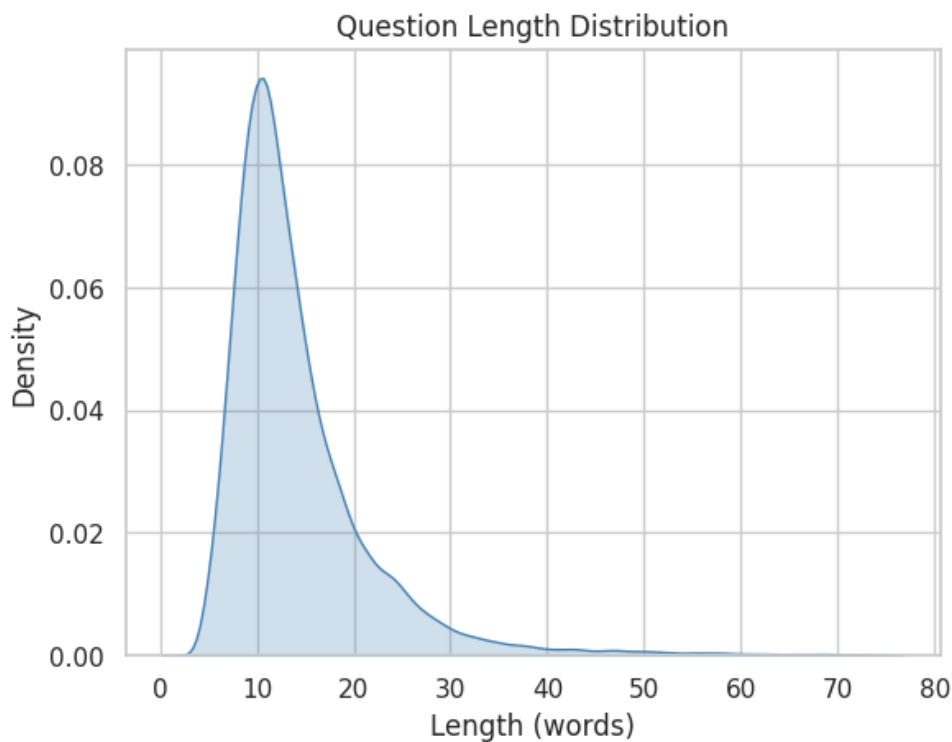
כפי שהיה מצופה, גם התפלגות מספר המילים הייחודיות בכל כתבה דומה להתפלגות גאוסית. השכיח הוא בקרוב 275 והממוצע 300. חשוב לציין כי חישוב מספר המילים הייחודיות כולל שמות וצורות שונות של אותה מילה.



איור 9: התפלגות היחס בין מספר המילים בכתבה למספר האסימונים לאחר tokenization.

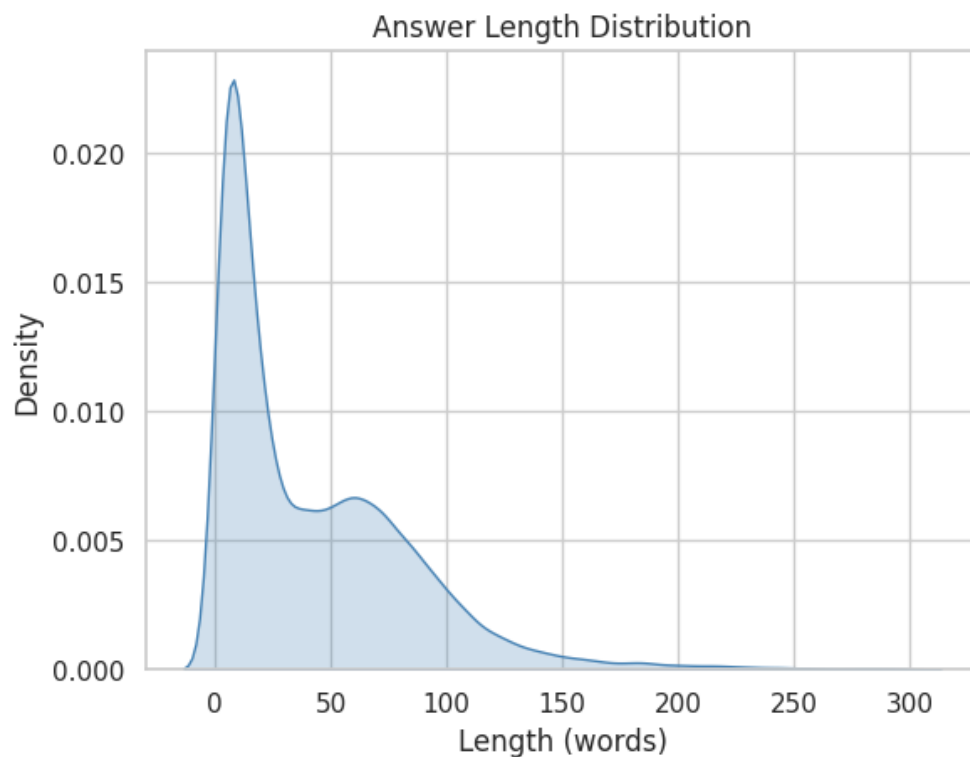
התפלגות היחס בין אורך הכתבה במילים לבין אורך הכתבה באסימונים היא כמעט צורת פעמון מושלמת של התפלגות גאוסית. מהגרף ניתן להסיק כי לוקח בממוצע כ-1.75 אסימונים כדי לייצג כל מילה בכתבה. במילים אחרות, כל אסימון מכיל 57% מהמשמעות הלשונית של מילה רגילה. סטית התקן מאד קטנה.

## מאגר המידע השני



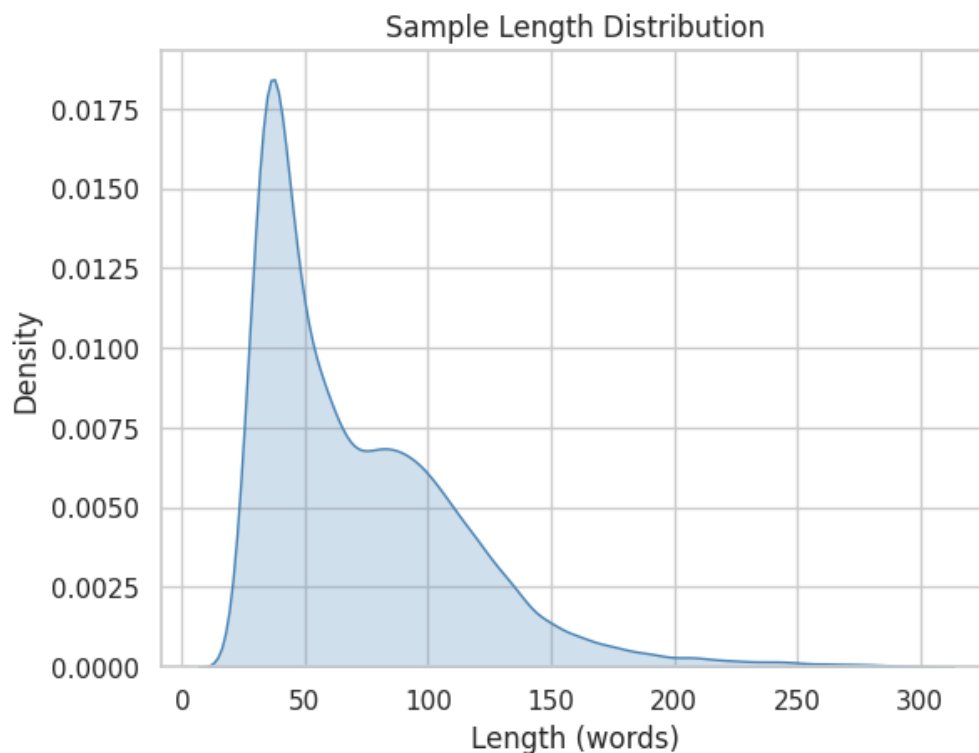
איור 10: התפלגות מספר המילים בכל שאלה.

התפלגות מספר המילים בכל שאלה מראה כי מרבית השאלות בעלות אורך של כ-11 מילים. יש מספר לא זניח של שאלות בעלות 20 ואף שלושים מילים. רובן המוחלט של השאלות שאני אספתי בעלות אורך פחות מ-20 מילים, לכן השאלות הארוכות נלקחו ממאגר הנתונים של אוניברסיטת סטנפורד.



איור 11: התפלגות מספר המילים בכל תשובה

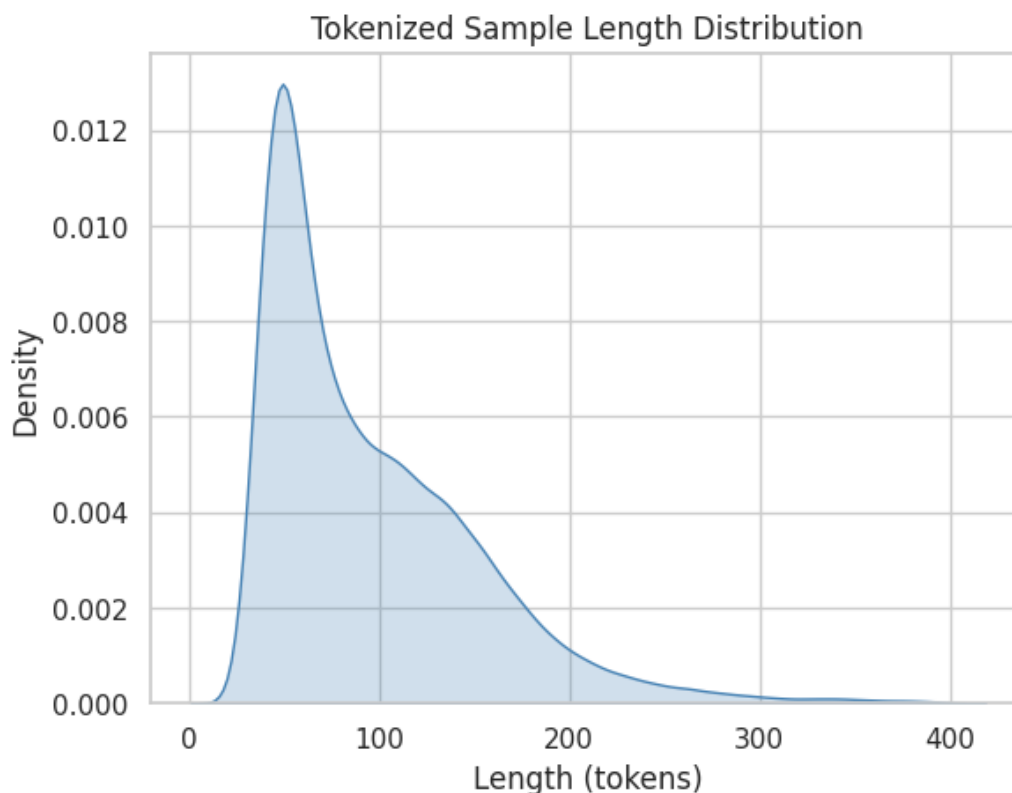
התפלגות מספר המילים בתשובות מזכירה צורת התפלגות bimodal, כלומר בעלת שתי פסגות. אני מאמין שהמודלים השפתיים שיצרו את המאגר, נוטים להשתמש באותה תבנית לתשובות. לכן, הרבה תשובות דומות מבנה ובמספר המילים. כך נוצרות שתי פסגות עיקריות של אורי התשובות.



איור 12: התפלגות מספר המילים בכל פריט במאגר הנתונים.

התפלגות זו היא למעשה חיבור שתי ההתפלגויות הקודמות ועוד קבוע מסויות, הנובע מתבנית הפריט. ניתן עדיין לראות את שתי הפסגות, אולם כעת הפסגה הימנית תלולה פחות. מרבית השאלות הן שאלות בעלות תושבה ישירה, למשל מה היא בירת ארצות הברית. אולם, חלק המשאלות פתוחות ואף דורשות הרבה כתיבה, כמו בקשה לכתוב מכתב יום הולדת לחבר.





איור 13: התפלגות מספר האסימונים בכל פריט במאגר הנתונים.

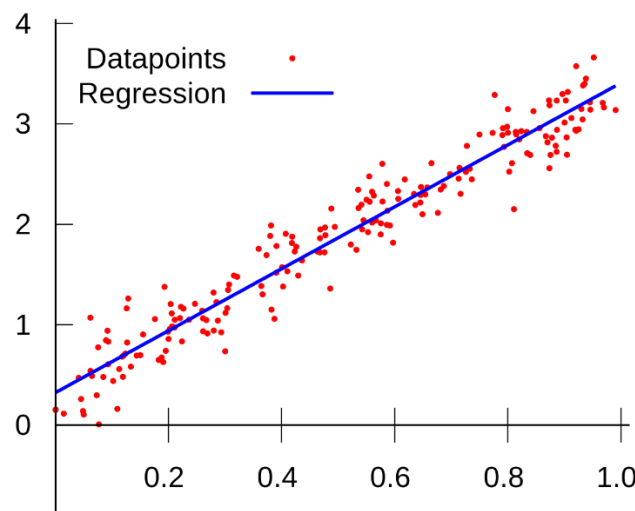
בעקבות אופי תהליך הקידוד של הפריטים ממילים לאסימונים, ההתפלגות של אורך הפריטים לפי מספר האסימונים כמעט זהה בצורתו להתפלגות אורכי הפריטים לפי מספר המילים. אוצר המילים הגדול של המקודד מאפשר יחס קידוד של 1.33 אסימונים לכל מילה. לכן, התפלגות הוסטה מעט ימינה. בנוסף, ניתן לראות כי מספר הפריטים בעלי אורך הגדול מ-512 הוא זניח, רק שלושה. על כן, אפשר למחוק אותם לשם יעילות אימון.

## שלב בניית ואימון המודל

### אבני היסוד של רשת נוירונים ולמידה עמוקה

#### רגרסיה לינארית

רגרסיה לינארית היא מודל סטטיסטי המאפשר מציאת קשר בין משתנים בהנחה שהוא לינארי, כלומר מהצורה  $y = m \times x + b$ . פעולה זו היא למעשה אלגוריתם הבינה המלאכותית הבסיסית ביותר, אשר יכולה למצוא קשר בין feature פשוטים. לדוגמה, באמצעות רגרסיה לינארית ניתן לנבא מחיר של נדל"ן בהינתן מספר החדרים או המ"ר שלו. חשוב לציין כי היכולת המתמטית הזו אינה ייחודית לחקר ופיתוח בינה מלאכותית, אלא משמש גם במגוון תחומים אחרים.



איור 14: דוגמה למציאת קו מגמה באמצעות רגרסיה לינארית. ניתן לראות שעל אף האקריות של המידע, קיים בין הנקודות קשר לינארי שיכול לקרב את ערכם של נקודות אחרות.

ניתן למצוא קשר בין מספר משתנים בלתי תלויים למשתנה תלוי אחד. צורתו המתמטית של קשר זה הוא  $y = m_1 \times x_1 + m_2 \times x_2 \dots m_n \times x_n + b$ . בהמשך לדוגמה הקודמת, אפשר לנבא את מחירו של נדל"ן בצורה מדויקת יותר באמצעות שימוש בכל המידע שקיים, כגון מספר החדרים, גודל המבנה, גודל השטח מספר השירותים וכו'.

ישנם מספר דרכים למצוא את המשוואה הלינארית המקיימת את הקשר בצורה הטובה ביותר. דרך אחת היא באמצעות נוסחה המוצאת את קו המגמה המדויק ביותר בהינתן מספר נקודות. דרך נוספת היא קרוב הפרמטרים של המשוואה באמצעות אלגוריתם רגרסיה.

$$\alpha = \frac{n \sum(xy) - \sum x \sum y}{n \sum x^2 - (\sum x)^2}$$

נוסחה 1: מציאת שיפוע של קו מגמה בהינתן אוסף נקודות.

$$\beta = \frac{\sum y - \alpha \sum x}{n}$$

נוסחה 2: מציאת הקבוע של קו מגמה בהינתן אוסף נקודות.

$$y = \alpha x + \beta$$

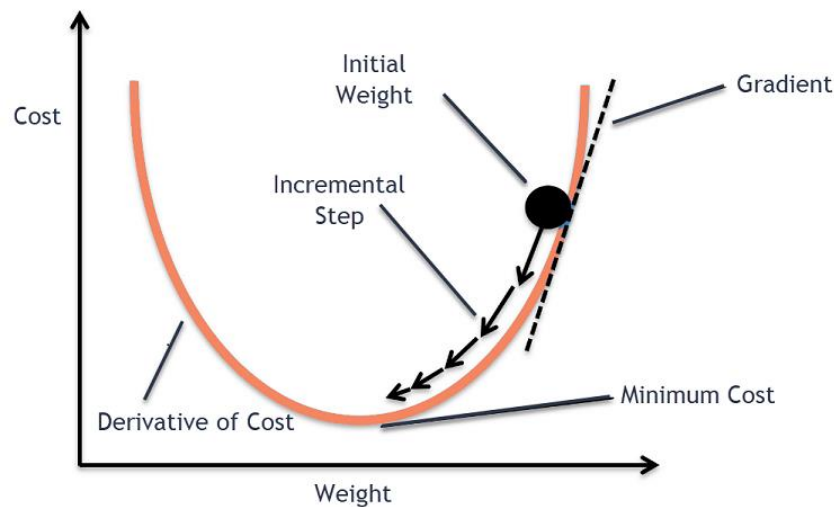
נוסחה 3: נוסחת קו מגמה לאחר שמחשבים את השיפוע והקבוע שלה.

הנוסחאות המתוארות לעיל מאפשרות מציאת השפיע  $\alpha$  והקבוע  $\beta$  היוצרים קו מגמה שמקרב בצורה הטובה ביותר את הקשר בין הנקודות הנתונות. עם זאת, כאשר מדובר בקשר בעל מספר פרמטרים בלתי תלויים, אי אפשר להסתפק בנוסחאות הנתונות. במקום, משתמשים בשיטת רגרסיה. שיטה זו מוצאת את קירובם של הפרמטרים באמצעות פונקציית מחיר המייצגת את דיוק המשוואה הליניארית.

$$Cost = \frac{1}{n} \sum_{i=0}^n (\hat{y} - y_i)^2$$

נוסחה 4: פונקציית מחיר מסוג Mean Square Error.

פונקציית המחיר סוכמת את ריבועי ההפרשים בין הערך המנובא  $\hat{y}$  לבין הערך האמיתי  $y$  של כל נקודה. כך, ככל שפונקציית המחיר יותר גדולה, דיוק המשוואה קטן יותר. אלגוריתם הרגרסיה מחפש לצמצם ככל האפשר את ערך המחיר של משוואת הקשר. הוא עושה זאת באמצעות התבוננות בנגזרות החלקיות של כל פרמטר, כלומר ההשפעה של כל פרמטר על ערך פונקציית המחיר. כאשר הנגזרת חיובית, משמע הגדלת הפרמטר יניב ערך מחיר גדול יותר, יש להקטין את הפרמטר. מנגד כאשר הנגזרת שלילית, הגדלת ערך הפרמטר תוביל לערך מחיר קטן יותר, ולכן צריך להקטין אותו.



איור 15: סקיצה של פונקציית מחיר. ניתן לראות שכאשר הנגזרת של פרמטר ביחס לפונקציית המחיר הינו חיובי, הקטנתו תוביל לערך מחיר קטן יותר. לעומת זאת, כאשר הנגזרת שלילית, הגדלת הפרמטר תוביל לערך פונקציית מחיר קטן יותר.

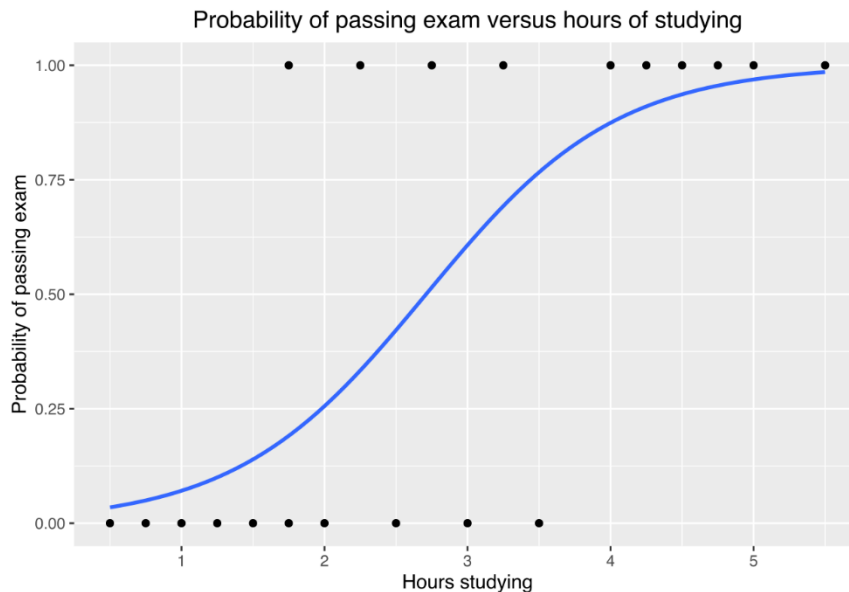
תהליך זה מכונה gradient descent ומטרתו להגיע לנקודת המינימום של פונקציית המחיר. מעבר על כל הנקודות האימון ועדכון הפרמטרים נקראה epoch. המתכנת יכול לקבוע את מספר הepochs שיעברו עד תום תהליך הלמידה. כדי לאפשר שליטה טובה יותר על התהליך נעשה שימוש בקבוע שנקראה learning rate. מספר זה קובע בכמה לשנות את ערכם של הפרמטרים בכל עדכון. כאשר הערך גדול מדי, ערכו של הפרמטר יכול "לקפוץ" מעל נקודת המינימום ולא להתכנס. לעומת זאת, כאשר הערך קטן מדי, ייקח לאלגוריתם זמן רב מדי להתכנס. האחריות לבחור ערך קצב למידה מתאים מוטלת על המתכנת.

```
def Gradient_Descent(x, y, learning_rate=0.001, epochs=1000):
    m = 0
    b = 0
    for i in range(epochs):
        for x, y in zip(x, y):
            y_hat = x * m + b
            loss = y_hat - y
            m = m - loss * x * learning_rate # derivative by m
            b = b - loss * learning_rate # derivative by b
    return (m, b)
```

לעיל מוצג יישום פשוט של אלגוריתם gradient descent בפיתון אשר מוצא משוואה לינארית המקרבת את הקשר בין המשתנים  $x$  ו- $y$ .

## רגרסיה אי-לינארית

רגרסיה לינארית הוא כלי עוצמתי למציאת קשר בין משתנים, אולם מספר רב של קרשים אינם לינאריים. הדוגמה הטובה ביותר לכך היא משימות הסבירות. לפי ההסכמות המתמטיות, הסתברות מוצגת כמספר הנע בין 0 ל-1. אם היינו מנסים לנבא את ההסתברות לגשם מחר באמצעות מודל לינארי, אנחנו עלולים לקבל הסברות שליליות או הסברות הגדולה מאחד.

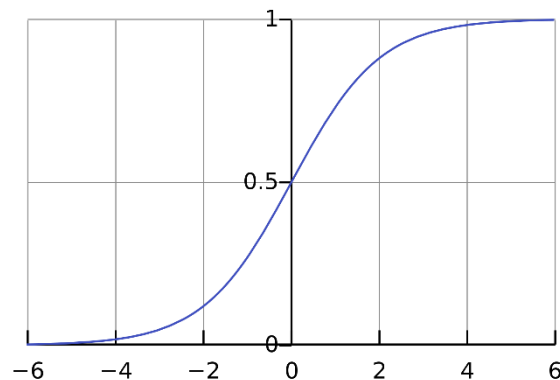


איור 16: דוגמה לפונקציית סיגמואיד המייצגת את הקשר בין מספר שעות הלמידה לבין ההסתברות לעבור את המבחן.

יש מספר פונקציות המשמשות לייצוג קשר אי לינארי. שתי הפונקציות החשובות והמשמעותיות ביותר בעולם הבינה המלאכותית הן סיגמואיד (sigmoid) ורלו (relu). פונקציית הסיגמואיד עוזרת למצוא קשרים סטטיסטיים, מכיוון שערכה תחום בין אפס לאחד. ניתן למצוא אותה לרוב בשכבה האחרונה של רשתות עם משימת סיווג, על אף שלעיתים היא גם משמשת בשכבות ביניים להוספת אי לינאריות.

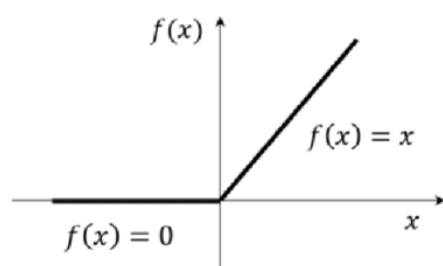
$$\hat{y} = \frac{1}{1 + e^{-z}}$$

נוסחה 5: נוסחה לפונקציית הסיגמואיד. הערך  $z$  מתקבל ממשווא לינארית כדי לאפשר שליטה בצורת הפונקציה.

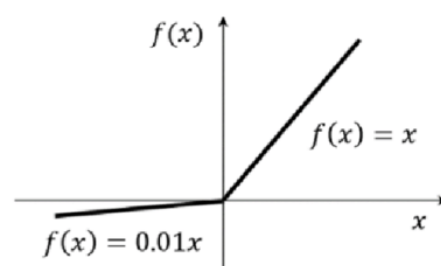


איור 17: סרטוט של פונקציית סיגמואיד.

פונקציית רלו פשוטה הרבה יותר מפונקציית הסיגמואיד, ולכן משתמשים בה בשכבות ביניים כדי להוסיף אי לינאריות. עובר ערכים קטנים מאפס, הפונקציה מחזירה אפס, כך גם הנגזרת של כל נקודה בצד השמאלי של ציר ה-y הינו אפס. הדבר עלול לגרום ל"נוירונים מתים" במודל הבינה המלאכותית. כלומר, נוירונים שאינם משנים את האותות בצורה מעולה ואף מוחקים את הנגזרת עבור שכבות קודמות. לכן, יש המשתמשים בגרסה מעט שונה של פונקציית הרלו המכונה LeakyReLU. פונקציה זו מוסיפה שיפוע בצידה השמאלי כדי לאפשר רגרסיה יעילה.



ReLU activation function



LeakyReLU activation function

איור 18: סרטוט של פונקציית רלו ו-LeakyReLU.

### למידה עמוקה

למידה עמוקה היא תת נושא בתחום הבינה המלאכותית שמתייחס לבניית ואימון מודלים גודלם מאוד. השימוש במילה "עמוק" נובע מהעובדה שהמודלים הללו כוללים מספר גדול של שכבות, וכל שמתקדמים בשכבות, הרשת מעמיקה את הפענוח וההבנה שלה. למשל, בתחום עיבוד תמונות, השכבה הראשונה עשויה לזהות קצבות של צורות ואף קווים. שכבות עמוקות יותר יכולות לפענח את

המידע ולדייק אותו עוד יותר. מקצבות וקווים אפשר להגיע לצורות, מצורות אפשר לזהות מרקם, ממרקם אפשר להסיק עצמים וכך האלה.

קיימים מספר סוגי אימון של רשת עמוקה, למידה מפוקחת ולמידה לא מפוקחת. בלמידה מפוקחת המתכנת מגדיר במפורש מה המודל צריך לנבא, כמו במשימת סיווג. המתכנת ממין מספר תמונות לקטגוריות, והמודל מצידו נדרש להבין כיצד הוא יוכל לסווג תמונות לקטגוריות הנכונות בלי לדעת אותן. לעומת זאת, בלמידה לא מפוקחת המודל צריך לזהות דפוסים וקשרים שאינם ידועים מראש. לדוגמה, חנות שרוצה לקבץ את הלקוחות שלה כדי להבין אילו סוגי לקוחות קיימים. המודל מקבל פרטים על הלקוחות, אבל לא את סוגי הלקוחות מכיוון שמידע זה אינו ידוע.

מספר השכבות הגדול מביא עמו מספר פרמטרים יוצא דופן. במודלים עמוקים, בניגוד למודלים אחרים, נדרש מאמץ מיוחד בזמן האימון. מודלים קטנים יכולים להתאמן על מעבד פשוט בלי להיתקל בבעיות זמן הרצה. אולם, מודלים בעלי מספר פרמטרים גודל צריכים לרתום את יכולות המחשוב של כרטיס מסך טוב, עם זיכרון מספק. מודלים עמוקים עוד יותר צריכים לרוץ על מספר מכשירים במקביל, לפעמים גם במספר מחשבים. כך למשל בפרויקט Open Pretrained Transformer, שממנו נלקח הבסיס למודל השני, נעשה שימוש בלא פחות מ-992 כרטיסי מסך מסוג A100 בעלי 80GB זיכרון. אימון המודל על מספר מכשירים במקביל מסבך את התהליך עוד יותר ונכתבו מאות מאמרים על שיטות שונות לעשות זאת.

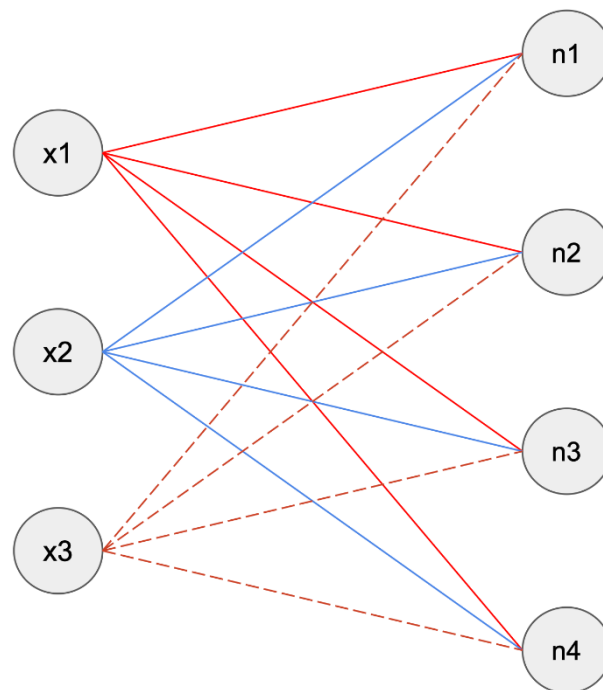
## סוגי שכבות

### שכבה מחוברת - dense layer

שכבת dense, המכונה גם fully connected layer, היא השכבה הבסיסית ביותר במודל בינה מלאכותית. היא מכילה רשימה חד-ממדית של פרספטרונים, אשר כל אחד מחובר לפלט השכבה הקודמת. כל פרספטרון מכיל מספר פרמטרים נלמדים כגודל הפלט של השכבה הקודמת. כל ערך בווקטור הפלט של השכבה הקודמת מקבל משקל ייחודי לו. הנוירונים המדומים הללו סוכמים את מכפלת הקלט בפרמטרים בהתאמה (פעולת dot product) ומוסיפים ערך נלמד המכונה bias. תוצאה זו עוברת פונקציית אקטיבציה שמוסיפה אי לינאריות. כל פרספטרון מחזיר ערך אחד, ולכן גודל פלט השכבה שווה למספר הפרספטרונים.

$$\hat{y} = activation(x \times kernel + bias)$$

נוסחה 6: נוסחת השכבה המחוברת.  $x$  הינו הפלט של השכבה הקודמת  $kernel$  הינו וקטור הפרמטרים של השכבה.



איור 19: דיאגרמה של שכבה מחוברת. ניתן לראות כי כל הפרספטרונים בשכבה השנייה מחוברים לכל התוצאות של השכבה הקודמת.

#### שכבת הטמעה - embedding layer

רשתות נוירונים מתקשות ללמוד לפענח ערכים דיסקרטיים. בניגוד לערכים רציפים, כמו גובה או משקל, המספר הסידורי של ערכים דיסקרטיים אינו מכיל מידע שימושי. כך לדוגמה אם נמספר את כל המילים באנגלית לפי הסדר האלפביתי, נקבל שהמילה במקום 1051 (apple) דומה יותר ל-1052 (appliance) מאשר ל-2171 (orange). לכן, כדי לאפשר לרשת הנוירונים ללמוד קשרים בין ערכים דיסקרטיים צריך למצוא דרך לייצג אותם בצורה בעלת משמעות.

ניתן ליצור מערך כאורך מספר הערכים, לאתחל את כל איבריו באפס, ולהפוך איבר אחד המייצג את האסימון הנתון ל-1. שיטה זו מכונה one hot encoding. היא עובדת עבור מספר מצומצם של ערכים, אולם מתגלות בעיות כאשר מנסים להפעילה על מספר גדול של ערכים. עם נסיה להשתמש בשיטה במשימת עיבוד שפה, נצטרך מערך גדול מאוד, שיוביל למספר פרמטרים אדיר. בנוסף, הקלט מכיל מעט מאוד מידע, רובו המוחלט מכיל אפסים, ורק איבר אחד מכיל מספר.

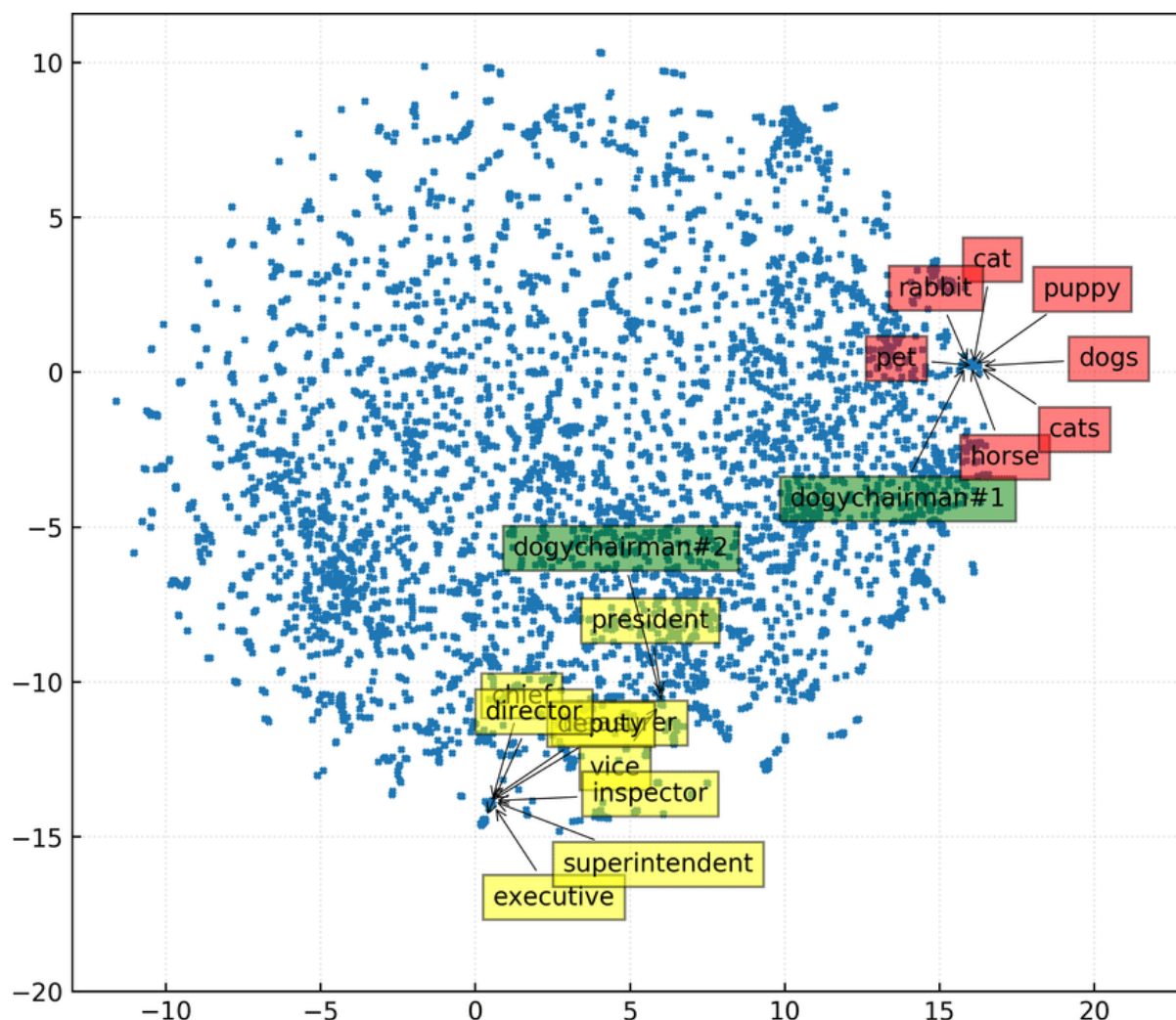


	1	2	3	4	5	6	7	8
I	1	0	0	0	0	0	0	0
ate	0	1	0	0	0	0	0	0
an	0	0	1	0	0	0	0	0
apple	0	0	0	1	0	0	0	0
and	0	0	0	0	1	0	0	0
played	0	0	0	0	0	1	0	0
the	0	0	0	0	0	0	1	0
piano	0	0	0	0	0	0	0	1

איור 20: דוגמה לקידוד משפט באמצעות שיטת one hot encoding. ניתן לראות כיצד כל מילה הופכת לווקטור באורך מספר המילים שבו רק ערך אחד דולק.

בשביל לקודד מספר גדול של ערכים דיסקרטיים הגו את שכבת ההטמעה, embedding layer, אשר פותרת את הבעיות של one hot encoding. היא ממפה כל ערך לווקטור בגודל  $d_{model}$  שמייצג את משמעותו. ניתן לחשוב על כל ערך בווקטור ההטמעה כדרוג של תכונה מסוימת. כך ווקטור ההטמעה הוא למעשה רשימת התכונות בעלי דרוג רציף שהמודל משוגל להבין. בהקשר של אסימונים, התכונות הללו יכולות להיות עד כמה האסימון מתאר דבר ממשי, האם האסימון בעל הקשר שמח, האם הוא בכלל מספר ולא מילה, וכדומה.

באמצעות ההטמעה המודל יכול לראות את האסימונים לא כערכים דיסקרטיים אלא ערכים הנמצאים במרחב כלשהו. אסימונים בעלי משמעות דומה יהיו קרובים אחד לשני, ואילו אסימונים בעלי משמעות שונה יהיו רחוקים. הם יכולים להיות דומים בתכונה אחת, אך שונים מאוד מבחינת תכונה אחרת. המודל יכול בקלות ללמוד כיצד להשתמש בערכי התכונות הללו כדי להסיק מסקנות לגבי משמעות האסימונים ולמצוא קשרים ביניהם.



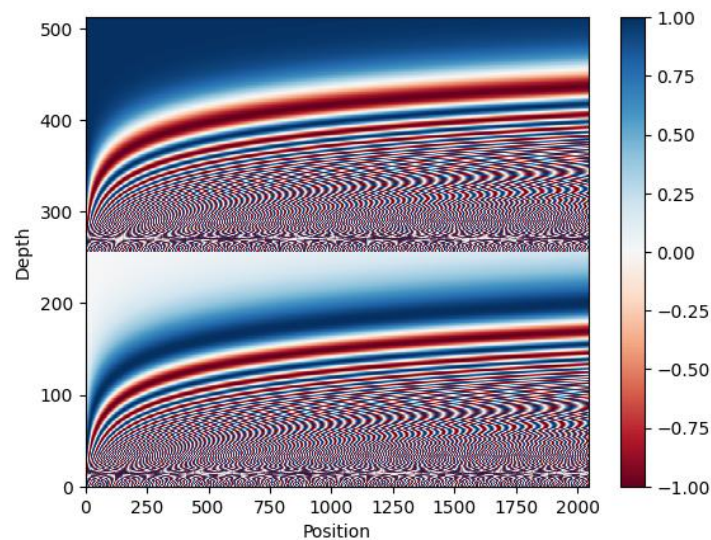
איור 21: איור המתאר את מרחב ההטמעה של מילים בעלי וקטור הטמעה בגודל שתיים. מובהק שמילים בעלות משמעות דומה קרובות אחת לשנייה ואף יוצרות מין קבוצות משמעות.

שכבמת ההטמעה מקנה לכל אסימון ווקטור הטמעה ייחודי באמצעות טבלת הטמעה. הטבלה מכילה את וקטור ההטמעה עבור כל ערך, והשכבה פשוט מחליפה כל אסימון בווקטור המתאר אותו. הערכים שנמצאים בווקטורים הללו הינם פרמטרים נלמדים. בתחילת האימון האסימונים מפוזרים במרחב בצורה אקראית. אולם, בתהליך הלמידה המודל מעדכן את ערכי ההטמעה בצורה שתגרום לביצועים טובים יותר. התוצאה היא וקטורים מדויקים שמכילים את המשמעות של כל אסימון. לו היו בסרטטים את מרחב ההטמעה, היו רואים שאסימונים דומים קרובים ואף יוצרים קבוצות של משמעות משותפת.

## שכבת קידוד מידע מיקומי - positional encoding layer

בניגוד למודלים שפתיים אחרים כגון CNN ו RNN, לטרנספורמר אין מידע בנוגע למקומו הסידורי של כל אסימון בקלט. מודלים מסוג זה מכונים bag of word, כיוון שהמודל מודע לאילו מילים הוא ראה, אך לא לסדרם במשפט, כמו שק המכיל מילים. על כן, צריך להוסיף מידע המתאר את מקומו של אסימון בקלט בכדי שהמודל יוכל לשפר את יכולות הפענוח שלו.

השיטה שנבחרה במסגרת פרויקט זה היא sinusoidal positional encoding המבוססת על פונקציות טריגונומטריות. עובר כל מיקום סידורי בטווח הקלט נקבע וקטור באורך  $d_{model}$  אשר מכיל מידע אודות התמצאותו במשפט. הוקטורים הללו נקבעים מראש ולא משתנים לאורך האימון, לכן הם לא נחשבים פרמטרים נלמדים. שכבת קידוד המידע המקומי מחברת בין הערכים שהיא מקבלת, וקטורי ההטמעה של האסימונים, לבין וקטורי המידע המקומי. כך עבור כל אסימון בקלט מתקבל וקטור המכיל את ההטמעה שלו ומיקומו הסידורי.



הנוסחאות איתן מחשבים את וקטורי המידע המקומי נבחרו בעקבות תכונת המתמטיות. ניתן להגיע מוקטור המיקום  $PE_{pos}$  למיקום  $PE_{pos+k}$  באמצעות מכפלה סקלרית. במילים אחרות, המיקום היחסי בין וקטורי המידע המקומי הוא לינארי. על כן, בזמן הלמידה העמוקה יהיה למודל קל יותר ללמוד הקשרים המבוססים על מיקומים יחסיים ולא סטטיים.

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

$$PE(pos, 2i + 1) = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

נוסחה 7: נוסחאות המידע המקומי של איבר. הנוסחה העליונה משמשת לעמקים זוגיים, והנוסחה השנייה משמשת לעומקים אי זוגיים.

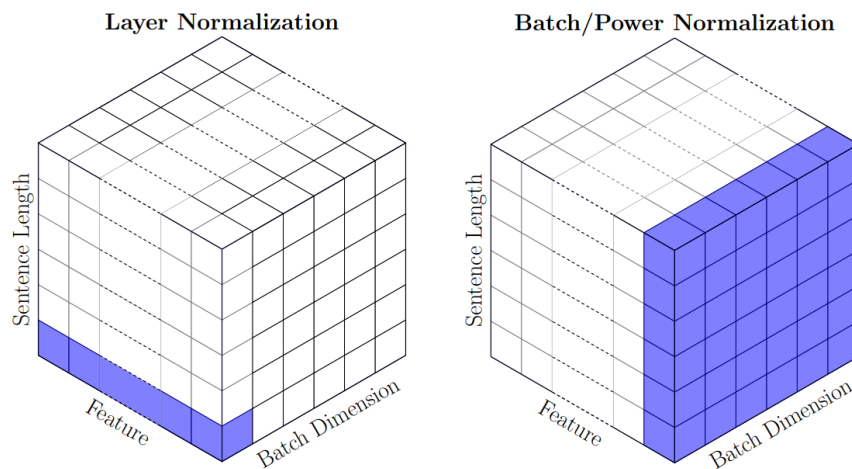
### שכבת נרמול - layer normalization

שכבת נרמול עוזרת לרשת נוירונים גדולה להתכנס ומונעת אי יציבות. השכבה מנרמלת את הערכים של כל feature כך שיהיו בעלי ממוצע קרוב ל0 וסטיית תקן קרובה ל1. טווח הערכים מונע בעיות אי יציבות בזמן אימון וinference שנובעות מערכים גדולים העוברים בין שכבות. בנוסף, הנרמול מוסיף "רעש" לקלט שמכריח את המודל למצוא קשרים ודפוסים כללים ולא להסתמך על פרטים ספציפיים, בדומה לשכבת dropout. שכבת הנרמול כוללת שני פרמטרים נלמדים,  $\gamma$  ו  $\beta$ , שנותנים לשכבה יותר שליטה על תהליך הנרמול ויכולים להשפיע על הממוצע וסטיית התקן במידת הצורך.

$$\hat{y} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \times \gamma + \beta$$

נוסחה 8: נוסחת שכבת נרמול, הערכים  $\gamma$  ו  $\beta$  הם פרמטרים נלמדים.

חשוב לציין כי לא מדובר על batch normalization, שהיא פעולת נרמול שונה. נרמול לפי שכבה מנרמל את הערכים של עבור כל feature, ואילו נרמול לפי קבוצה מנרמל את הערכים עבור קבוצת אימון. ברשתות NLP נעשה שימוש בנרמול לפי שכבה מכיוון שהקלט יכול להיות בגדלים שונים. בעוד שמספר הערכים של כל feature תמיד קבוע ולכן ניתן לנרמל אותם.



איור 22: ההבדל בין נרמול לפי שכבה ונרמול לפי קבוצה. ניתן לראות שבנרמול לפי שכבה ערכי feature בלבד מנורמלים, ואילו בנרמול לפי קבוצה, ערכי הקלט של הרשת עצמה מנורמלים.

### שכבת masking

שכבת masking מאפשרת למודל להוסיף "תגים" על חלקים מסוימים של הקלט כלי ששכבות ידעו להתייחס עליהם בהתאם. כך אסימוני ריפוד מקבלים תג שאומר לשכבות להתעלם מהם, מכיוון שהם לא מכילים מידע שימושי. שכבות התשומת לב יודעות שלא להקנות להם חשיבות הקשרית, ונוסחות ההפסד ומחיר יודעות שאין חשיבות לפלט הפונקציה עבור ערכים אלה.

### שכבת softmax

שכבת softmax היא למעשה פונקציה שמדנה את פונקציית המקסימום, אך בצורה גזירה. כאשר בונים רשתות נוירונים, לעיתים רוצים פונקציה שבוחרת את האיבר בעל הערך הגדול ביותר. למשל, בשכבה האחרונה של מודל סיווג, שבו כל ערך בתוצאה מייצג את ההסתברות של הקטגוריה המתאימה לו. אילו היו מוסיפים פונקציית מקסימום רגילה, התוצאה לא תהיה גזירה. ובכך לא יהיה ניתן לערוך את תהליך החלחול לאחור והמודל לא ילמד.

על כן, נוצרה פונקציית softmax שמדמה את הלוגיקה של פונקציית המקסימום הרגילה, אך בצורה גזירה. הפונקציה לא מקנה לאיבר הגדול ביותר הסתברות של 1, אלא מספר קרוב לו. איברים אחרים מקבלים הסברות קטנה שלא תשפיעה באופן דרמטי על תאוצות הרשת. כדי לקבל את הסתברות

הsoftmax של איבר, מעלים אותו למעריך של  $e$ , ומחלקים בסכום כל האיברים כאשר מעלים אותם למעריך של  $e$ .

$$\sigma(\vec{x})_i = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$$

נוסחה 9: הנוסחה לחישוב ההסתברות של איבר  $\sigma(\vec{x})_i$  באמצעות פונקציית softmax.

$$\Sigma \left( \begin{bmatrix} 1 \\ 2 \\ 8 \end{bmatrix} \right) = \begin{bmatrix} 0.001 \\ 0.002 \\ 0.997 \end{bmatrix}$$

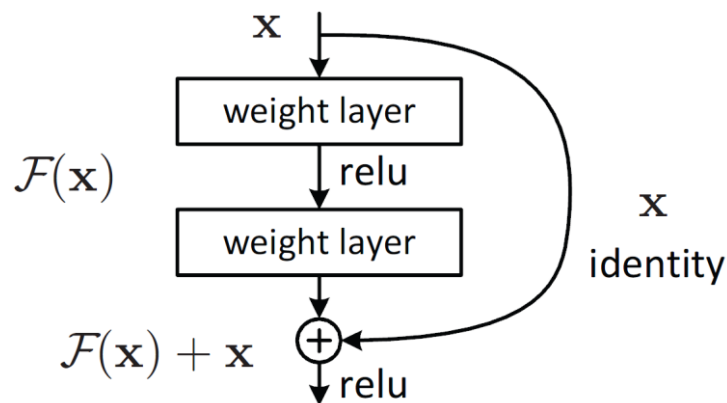
נוסחה 10: דוגמה לפעולת softmax על קבוצת מספרים.

מכיוון שסכום ההסתברויות לאחר הפעלת פונקציית softmax היא 1, משתמשים בפונקציה למטרות שונות שלא למציאת האיבר הגדול ביותר. למשל, נעשה שימוש בפונקציה כדי לנרמל ערכים או לפני שצריכים לעשות פעולת dotproduct לפי הסתברויות.

### שכבת residual connection

בעיה שפוקדת רשתות נוירונים עמוקות היא דילול הנגזרת. בתהליך החלחול לאחר השכבות שנמצאות קרוב לסוף הרשת מקבלות ערכי נגזרת גדולים. אולם, כל שכבה מדללת את ערכי הנגזר, ואחרי מספר שכבות הנגזרת כה קטנה שהפרמטרים לא משתנים בצורה משמעותית. כך נוצר מצב שבו רק השכבות שקרובות לסוף הרשת לומדות, ואילו שכבות שקרובות לתחילת הרשת נותרות ללא שינוי משמעותי ממצבן ההתחלתי.

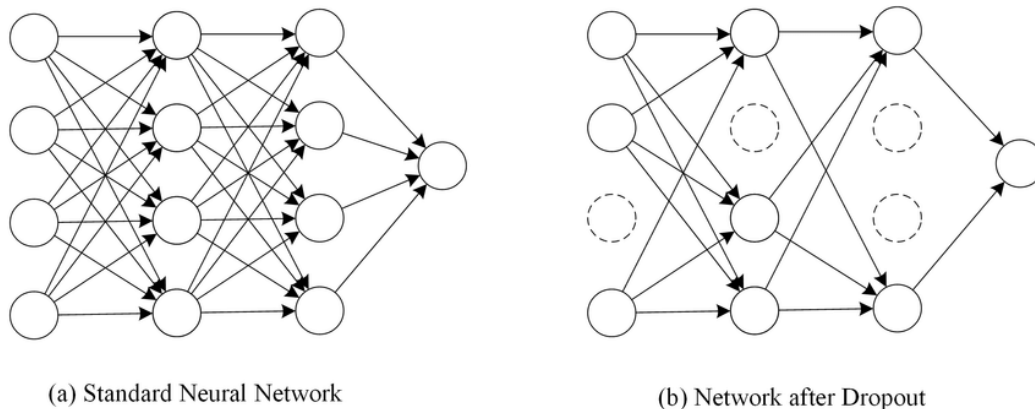
פתרון מאוד יעיל הוא שימוש בחיבורים ישירים, residual connections, שמדלגים על שכבות שלמות כדי שערכי הנגזרת לא יושפעו בידיהן. כך חיבורים אלה משמשים כעורקים שמאפשרים לכל שכבות הרשת להשתמש בנגזרת בתהליך החלחול לאחר. בנוסף, חיבורים אלה מכריחים את הרשת למצוא קרשים ודפוסים כללים ולא להקנות חשיבות גדולה מדי לאיברים בודדים, בדומה לשכבת dropout.



איור 23: סרטוט של חיבור ישיר, *residual connection*. ניתן לראות כיצד החיבור מדלג על שכבות וכך מביא ערכי נגזרת לכל גם לשכבות שנמצאות קרוב לתחילת הרשת.

### שכבת dropout

כאשר מאמנים רשתות נוירונים עמוקות, הוספת שכבות dropout יכולה לשפר את ביצועי המודל ולמנוע *overfitting*. שכבת dropout מכבה נוירונים בצורה אקראית בזמן האימון. כך הרשת לא יכולה לשנן את הדוגמאות במאגר המידע, ולכן אין חשש ל'*overfitting*'. בנוסף, כיבוי חלק מהנוירונים מאלץ את הרשת למצוא פתרון כללי ולא להסתמך על פרטים ספציפיים בקלט. נהוג להוסיף שכבות dropout לפני שכבות רבי פרמטרים כמו שכבת *dense*.



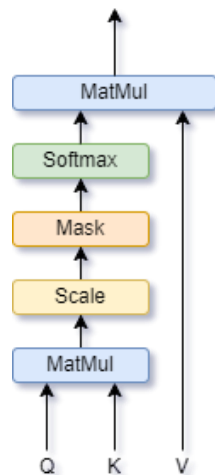
איור 24: דוגמה לרשת נוירונים פשוטה לפני ואחרי שימוש בשכבת dropout. ניתן לראות כיצד חלק מהנוירונים ברשת dropout כבויים ואינם מעבירים אותות.

בתחילת כל צעד אימון, חלחול קדימה ואחורה, שכבות dropout בוחרות בצורה אקראית *rate* אחוז מהנוירונים, ומכבות אותם. כדי לשמור על עוצמת האותות השכבה כופלת את ערכו של כל נוירון שלא

נכבה בקבוע  $\frac{1}{1-rate}$ . כאשר משתמשים ברשת שלא למטרת אימון שכבת dropout לא פועלת כלל ואינה משפיע על תוצאות המודל.

### שכבת תשומת לב עצמי - masked self-attention

שכבת התשומת לב היא הבסיס של מודל הטרנספורמר והחלק שמבדיל אותו ממודלים שפתיים אחרים. באמאר המהפכני שיצא לאור ב-2017 בשם "Attention is all you need" הכותבים הדגימו כיצד ניתן לבנות מודל שפה יצרני באמצעות מנגנוני תשומת לב בלבד. עד להוצאת המאמר, רשתות NLP השתמשו במודל רקורסיבי RNN או קונבולוציה CNN. הארכיטקטורות הללו סבלו משכחה וטווח הקשר מוגבל, בעיות ששכבות תשומת לב פותרות.



איור 25: דיאגרמה המתארת את מבנה שכבת התשומת לב.

בצורתו המופשטת ביותר של מנגנון התשומת לב, ניתן לדמיין את מילות הקלט יושבות בשולחן ארוך ובתורן שואלות שאלות. הן מקשיבות לתשובות של שאר המילים ומדייקות את משמעותן. לדוגמה, בתורה של המילה "bank", היא תבדוק האם יש בשולחן מילים כמו "money", "accounts", ו"transactions". כך המילה תבין שהיא בנק כספים. לעומת זאת, אם היא מגלה שבשולחן יושבות מילים כמו "river" או "stream", המילה תבין שמשמעותה גדת נהר.

מנגנון התשומת לב משתמש בשלושה מטריצות כדי לחשב את הקשר בין ערכים ולעדכן אותם בהתאם. ניתן לחשוב על המנגנון כמנוע חיפוש. מטריצת Queries מכילה את ה"בקשה" של כל פריט בקלט, הטקסט שמכניסים למנוע החיפוש; מטריצת Keys מכילה את ה"כותרת" של כל פריט בקלט;

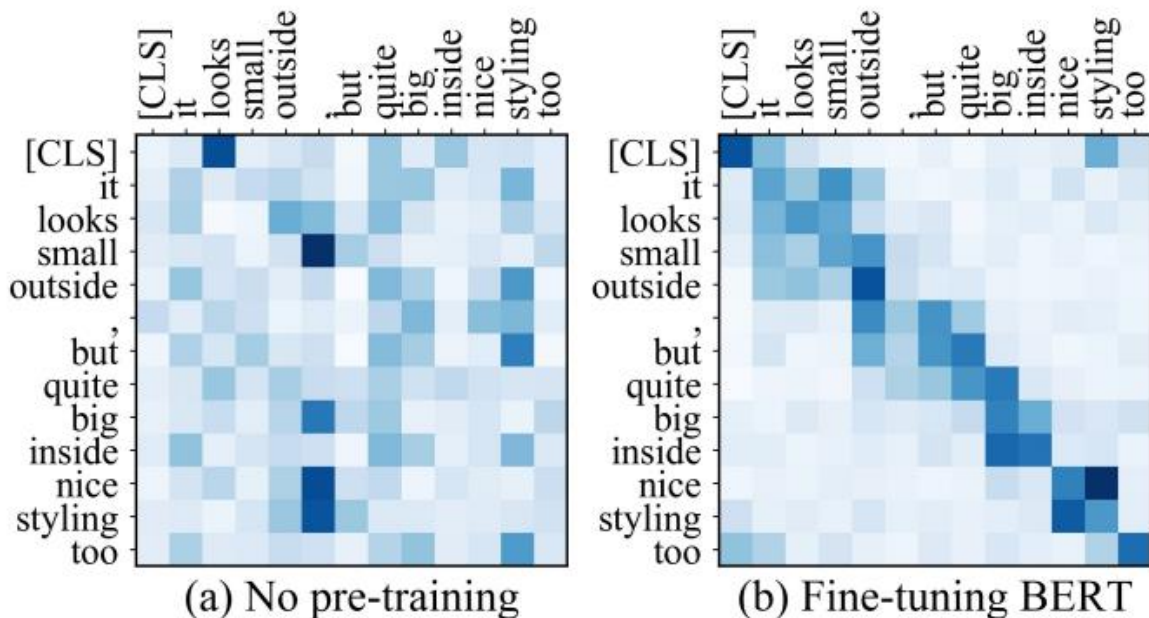


ומטריצת Values מכילה את הערך של כל פריט. ערכים אלה מתקבלים על ידי מכפלת קלט השכבה בשלוש מטריצות. מטריצות אלה מכילות את הפרמטרים הנלמדים שאותם המודל מדייק בזמן האימון.

$$\text{Affinity}(Q, K) = \frac{QK^T}{\sqrt{d_k}}$$

נוסחה 11: חישוב מטריצת הזיקה של פרטי הקלט באמצעות מטריצות Keysi Queries.

בשלב הראשון של מנגנון התשומת לב צריך לחשב את הזיקה בין מטריצות Keysi Queries. במילים אחרות, אליו מפתחות קשורות לאילו בקשות ובכמה. ישנם מספר דרכים לעשות זאת, במאמר נעשה שימוש בפעולת dot product. מחלקים לפי ערך  $\sqrt{d_k}$ , שורש גודל מטריצות Keysi Queries, מכיוון שמטריצות יותר גדולות מובילות לערך dot product גדול יותר. התוצאה היא מטריצה בגודל אורך הקלט על אורך הקלט שמייצגת את הזיקה בין פרטי הקלט.



איור 26: דוגמה למטריצת זיקה לפי ואחרי אימון. ניתן לראות כי לפני האימון ערכי הזיקה אקראיים. לעומת זאת, אחרי אימון ערכי הזיקה בין מילים קשורות ברורה: *it-small*, *quite-big*, *big-inside*, *nice-styling*.

מאחר ומטרת המודל בתהליך הלמידה הוא לנבא את האסימון הבא, צריך להסתיר אותו. מעבר לכך, בתהליך יצירת התוכן המודל בונה את הטקסט מילה-מילה, ולכן אין למילים ראייה קדימה, כיוון שאלו עוד לא קיימים. על כן, צריך למחוק את ערכי הזיקה של כל מילה עם כל המילים שבאות אחריה. לשם כך נשתמש בmask מתאים.

$$M = \begin{pmatrix} 0 & -\infty & -\infty & -\infty & -\infty \\ 0 & 0 & -\infty & -\infty & -\infty \\ 0 & 0 & \ddots & -\infty & -\infty \\ 0 & 0 & 0 & 0 & -\infty \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

נוסחה 12: ערך mask שמסתיר את האסימונים הבאים במטריצת הזיקה.

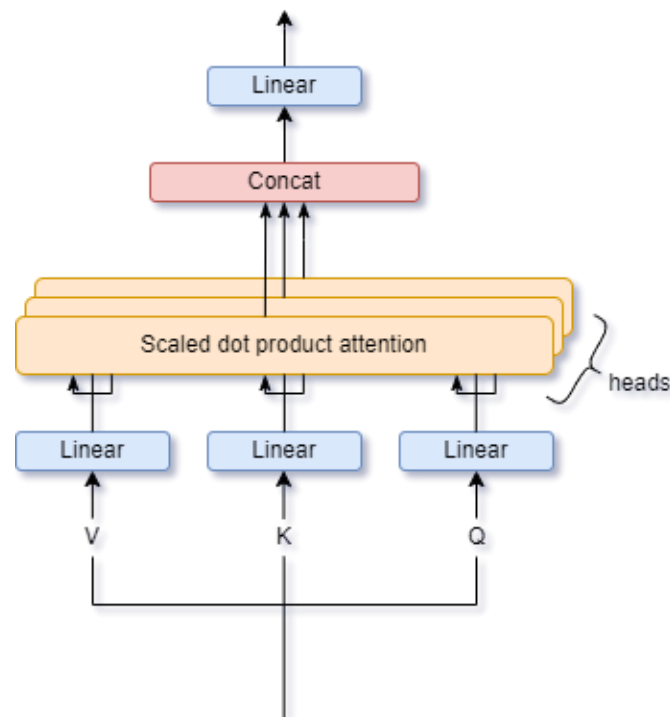
כדי לדייק את ערכי האסימונים יש לעדכן אותם לפי ערכי במטריצת ה-Values. מטריצת הזיקה מייצג את עוצמת הקשר בין איברי הקלט, ולכן נשתמש בה בכדי לדעת איזה משקל להקנות לכל ערך ב-Values. יש לנרמל את מטריצת הזיקה באמצעות פונקציית softmax כך שסכום כל שורה יהיה אחד. לאחר מכן נכפול את מטריצת הזיקה במטריצת ה-Values, כך שכל שערך הזיקה בין שני אסימונים גדול יותר, הם ישפיעו אחד על השני יותר.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}} + M\right) \times V$$

נוסחה 13: נוסחת תשומת לב עצמי מלאה.

#### שכבת תשומת לב רב-ראשית - multi-head masked self-attention

כותבי המאמר "Attention is all you need" הציעו לשפר את יכולת הלמידה של מנגנון התשומת לב מאמצעות ארכיטקטורה רב-ראשית. כל איבר בקלט צריך לברר הרבה דברים על שאר האיברים, הוא צריך לדעת מי הנושא, מי הנשוא, את המבנה התחבירי של המשפט, אילו מילות תיאור יש, את הנימה הכללית של הטקסט וכדומה. פעולה אחת של תשומת לב לא יכולה לספק את כל המידע במקביל.



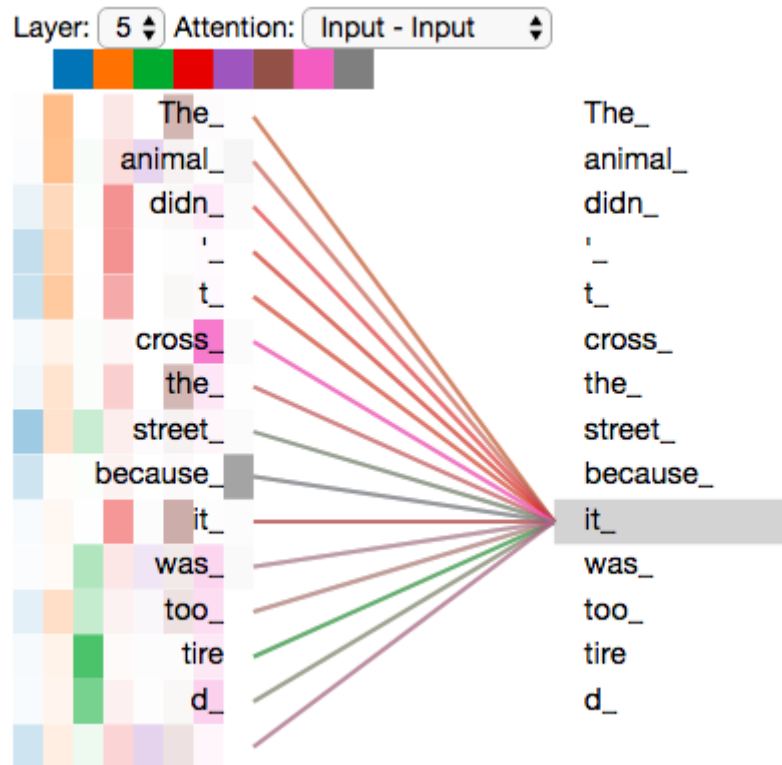
איור 27: תיאור של ארכיטקטורת תשומת לב רב-ראשית. ניתן לראות כיצד הקלט מחולק למספר ראשים ולאחר מכן משורשר.

כזכור מההסבר על שכבת ההטמעה, אפשר לדמות כל ממד הטמעה לדרוג של תכונה מסוימת אצל האסימון. מכאן נובע שמרבית החיפוש בתשומת הלב נעשה בתוך אותו ממד הטמעה. אפשר, אם כן, לחלק את וקטור ההטמעה של איברי הקלט לווקטורים בעלי מספר ממדים קטן יותר, ולעשות את פעולת תשומת הלב על הקבוצות הללו. כל קבוצת וקטורים שעוברים פעולת תשומת לב מכונים ראש אחד. לבסוף, השכבה משרשרת את תוצאות הראשים ומכפילה אותה במטריצה שמרכיבה מחדש את ההטמעה של כל אסימון.

$$\text{MultiHeadAttention}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \dots, \text{head}_h)$$

$$\text{where: } \text{head}_i = \text{Attention}(Q_i, K_i, V_i)$$

נוסחה 14: ייצוג מתמטי של חלוקת קלט השכבה למספר ראשי תשומת לב ושרשורם.



איור 28: איור של עוצמת הזיקה בין אסימונים בתשומת לב רב ראשית. כל בצע מייצג ראש תשומת לב שונה. ניתן לראות כי הראשים מתמקדים על חלקים שונים של המשפט, מכיוון שהם מחפשים קשרים שונים.

### שכבת חלחול קדימה - feed forward layer

במודל נעשה שימוש בשכבת חלחול קדימה בנוסף למנגנון התשומת לב כדי לדייק את משמעותם של האסימונים. שכבה זו מורכבת משני שכבות dense. השכבה הראשונה הינה בגודל  $d_{ff}$  ומשתמשת בפונקציית אקטיבציה רלו רגיל. השכבה השנייה לא משתמשת בפונקציית אקטיבציה וגודלה כגודל ממד ההטמעה,  $d_{model}$ .

$$FFN(x) = \max(0, xW_1 + b_1) \times W_2 + b_2$$

נוסחה 15: נוסחת שכבת החלחול קדימה. פונקציית רלו ניתנת להצגה כ- $\max(0, x)$ .

במאמר הפורץ דרך "Attention is all you need" הכותבים הסבירו והדגימו כיצד ניתן ללכד את שכבות התשומת לב, חלחול קדימה, נרמול וחיבור ישיר ליחידה אחת יעילה בשם הטרנספורמר. במבט על, המודל מקבל כקלט אסימונים מוטמעים, מדייק את משמעותם הלשונית לפי הקשר הקטע ומפענח אותם. תוצאת הטרנספורמר הוא וקטור אסימונים עם הטמעה מדויקת יותר שמכילה את המשמעות הלשונית, התחבירית והסמנטית שלהם.

The diagram illustrates the Transformer architecture, divided into an **Encoder** (left) and a **Decoder** (right).

**Encoder:**

- Inputs** are processed by **Input Embedding** and **Positional Encoding** (represented by a sine wave icon).
- The encoder consists of  $N \times$  identical layers. Each layer contains:
  - Multi-Head Attention** (orange box): **Encoder self-attention**; tokens look at each other. queries, keys, values are computed from encoder states.
  - Add & Norm** (green box): **Encoder self-attention**; tokens look at each other. queries, keys, values are computed from encoder states.
  - Feed Forward** (blue box): **Feed-forward network**; after taking information from other tokens, take a moment to think and process this information.
  - Add & Norm** (green box): **Feed-forward network**; after taking information from other tokens, take a moment to think and process this information.
- Residual connections and layer normalization** (orange dashed lines) connect the input of each layer to the output of the layer.

**Decoder:**

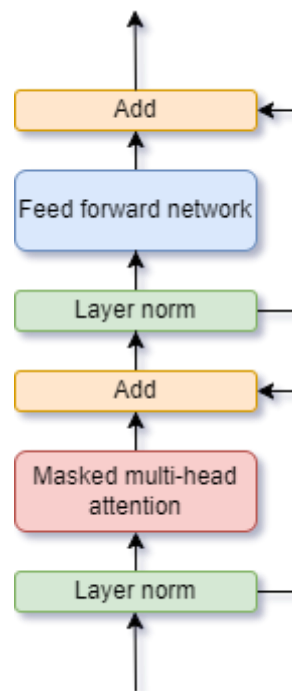
- Outputs (shifted right)** are processed by **Output Embedding** and **Positional Encoding** (represented by a sine wave icon).
- The decoder consists of  $N \times$  identical layers. Each layer contains:
  - Masked Multi-Head Attention** (orange box): **Decoder self-attention (masked)**; tokens look at the previous tokens. queries, keys, values are computed from decoder states.
  - Add & Norm** (green box): **Decoder self-attention (masked)**; tokens look at the previous tokens. queries, keys, values are computed from decoder states.
  - Multi-Head Attention** (orange box): **Decoder-encoder attention**; target token looks at the source queries – from decoder states; keys and values from encoder states.
  - Add & Norm** (green box): **Decoder-encoder attention**; target token looks at the source queries – from decoder states; keys and values from encoder states.
  - Feed Forward** (blue box): **Feed-forward network**; after taking information from other tokens, take a moment to think and process this information.
  - Add & Norm** (green box): **Feed-forward network**; after taking information from other tokens, take a moment to think and process this information.
- Residual connections and layer normalization** (orange dashed lines) connect the input of each layer to the output of the layer.

**Output:**

- The final output is processed by a **Linear** layer and a **Softmax** layer to produce **Output Probabilities**.

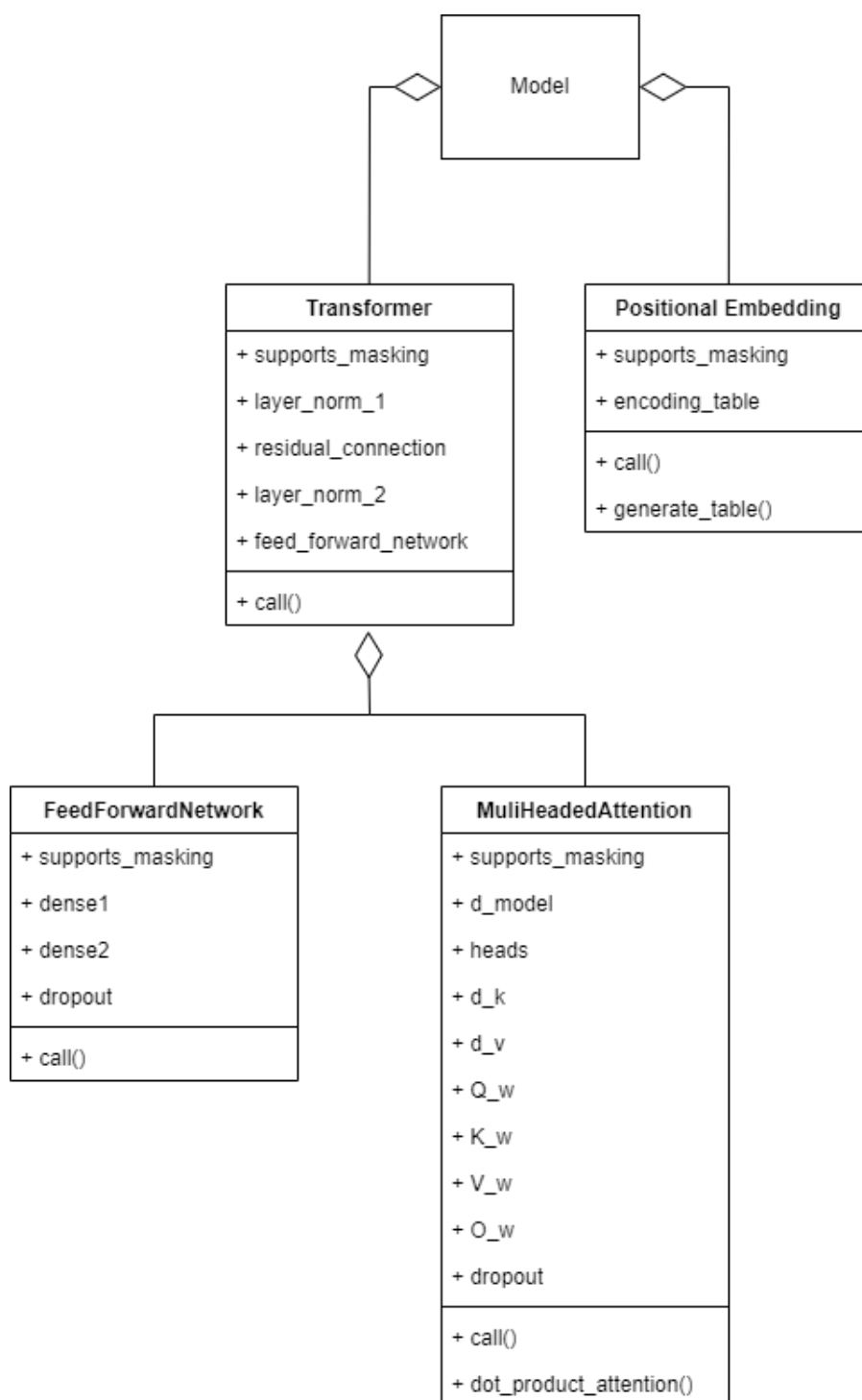
ארכיטקטורת הטרנספורמר שנמצא במודלים שנבנו במסגרת פרויקט זה שונה מעט מהמודל המקורי שהוצג במאמר. מטרת הכותבים הייתה לבנות רשת נוירונים שמשוגלת לתרגם קטעים בין שפות. לכן הם השתמשו הן ביחידת טרנספורמר מפענחת והן ביחידות טרנספורמר המקודדת. המקודד מקבל כקלט את הקטע בשפה המקורית ויוצר ממנו ייצוג לשוני שהרשת יכולה להבין. המפענח מקבל כקלט את המשפט בשפה החדש, או יותר נכון את ההתחלה שלו, ופענח אותו כדי לנבא אסימונים נוספים בתהליך autoregressive.

מטרתו של המודל שנבנה בפרויקט זה היא ליצור משפטים בתהליך autoregressive, ולכן נעשה שימוש רק ביחידת הפענוח. ומכיוון שלא נעשה שימוש ביחידת הקידוד, החיבור ביניהם באמצעות cross-attention הוסר. יתר על כן, בטרנספורמר המקורי שכבות הנרמול והחיבור ישיר נמצאים אחרי שכבת התשומת לב והחלחול קדימה. אחרי ניסוי וטעיה נתגלה כי העברת שכבת הנרמול לפני שכבות התשומת לב וחלחול קדימה מניבה תוצאות טובות יותר.



איור 30: ארכיטקטורת הטרנספורמר שנבנה במסגרת פרויקט זה. המודל מבוסס על יחידת הטרנספורמר המפענחת שהוצג במאמר "Attention is all you need" עם שינוי קטן למקום שכבות הנרמול והחיבור ישיר.

## תיאור UML של המחלקות



איור 31: דיאגרמת UML של מחלקות הרשת שמומשו. יש לציין כי שכבות אחרות שלא מומשו לא נכללו בתיאור זה.

## סמפלר

כדי להשתמש במודל יצרני נדרש אלגוריתם שיבחר אסימון מתוך התפלגות ההסתברויות שחזתה רשת הניורונים. לו היינו בוחרים באסימון בעל ההסתברות הגבוהה ביותר כל פעם, המודל היה נוטה ליצור תוכן לא יצרתי ולעיתים הוא יכנס ללולאת רצף אסימונים ממנה הוא לא יכול לצאת. לכן יש צורך באלגוריתם שידע לבחור את האסימון הבא כצורה שמתאימה לתוכן שהמודל התבקש ליצור ולמנוע חזרה על אותם המילים.

לאלגוריתם כזה קוראים סמפלר (sampler) מכיוון שהוא דוגם אסימונים מן התפלגות ההסתברויות שמבא המודל. יש מספר אלגוריתמים שיכולים לסייע, החלטתי להשתמש באחד בשם `top_p`. הסמפלר קודם כל ממין את ההסתברויות מהגדול לקטן. לאחר מכן הוא בוחר את האסימונים שסכומם לא עולה על הסתברות  $p$  שנקבעה מראש. לבסוף הוא בוחר אסימון אחד מהרשימה. כך אפשר לשלוט על מספר האסימונים מהם הסמפלר בוחר.

בנוסף ניתן לחלק את logits לפני פעולת softmax במספר כלשהו המכונה `temperature`. הפעולה למשעה שולטת על צורת ההתפלגות, יותר אחידה או יותר מוקצנת. ערכים הגדולים מ-1 של `temperature` הופכים את ההתפלגות ליותר אחידה, ואילו ערכים הקטנים מ-1 מקצינים את ההסתברויות הגדולות.

באמצעות שני המשתנים `temperature` ו-`top_p` המשתמש יכול לשלוט על האקריות של המודל וב"ביטחון העצמי" שלו. לתכנים שונים נדרשים ערכים שונים. למשל, אם שואלים את המודל השיחתי שאלה שתשובתה צריכה להיות עובדתית, נרצה אקריות נמוכה וביטחון עצמי גבוהה. מנגד, אם נרצה שהמודל יכתוב סיפור יצירתי נבחר בערכים שיובילו לאקריות גבוהה וביטחון עצמי בינוני.

## המודלים שעליהם בוצע האימון

כפי שהוסבר במבוא, במסגרת פרויקט זה אומנו שני מודלים. המודל הראשון נבנה מאפס והתאמן על מאגר נתונים המורכב מכתובת עיתון. המודל השני הוא מודל קיים מטעם פרויקט Open Pretrained Transformers, שמבוסס על אותה ארכיטקטורה כמו המודל הראשון, אך גדול פי 12.5. המודל השני כבר עבר את האימון הראשוני, ולכן בפרויקט זה כולל את האימון השני שהופך אותו לבינה מלאכותית שחתית בלבד.

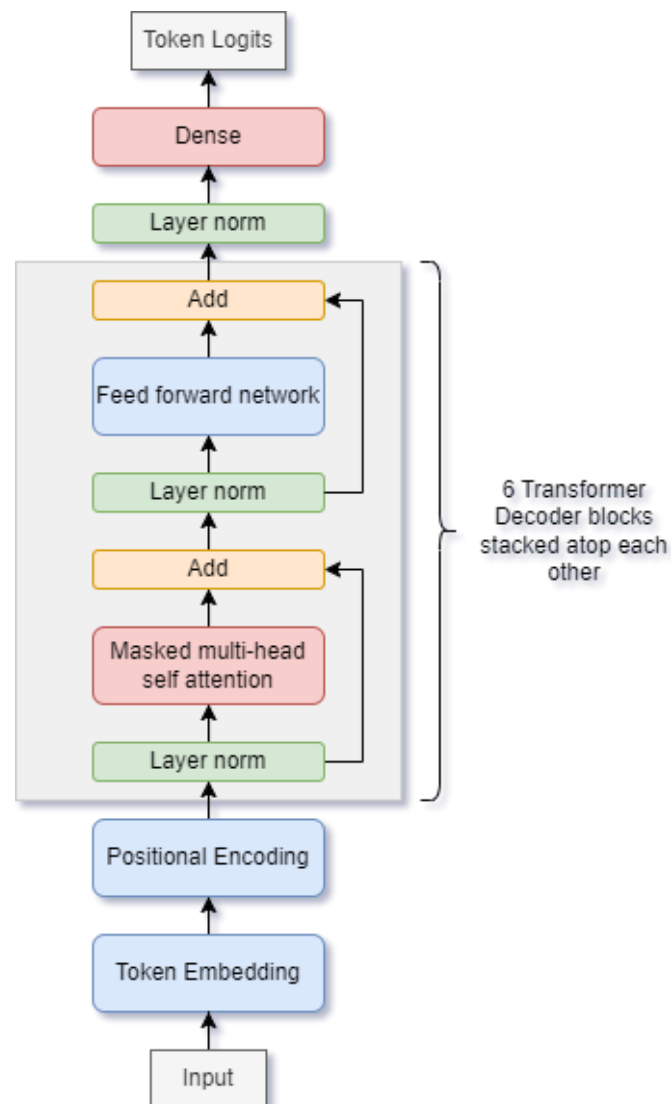


## המודל הראשון

ניתן לחלק את המודל הראשון לשלושה חלקים עיקריים: הטמעה, פענוח והסקה. החלק הראשון מורכב משכבת הטמעה ושכבת קידוד מיקום. מטרת החלק הראשון היא לייצג את האסימונים באופן שרשת הנוירונים יכולה להבין ולעבד. רק שכבת ההטמעה מכילה פרמטרים נלמדים, שהם וקטור ההטמעה של כל אסימון. לעומת זאת, ערכי שכבת קידוד המקום מחושבים מראש. חלק זה מקבל מערך בגודל 256 והופך אותו למטריצה של 256 על 384, כאשר 384 הוא ממד ההטמעה.

החלק השני הוא פענוח האסימונים. כעת לרשת יש ייצוג של הקלט שהיא מבינה, ועליה לעבד ולפענח אותו כדי שתוכל להסיק מסקנות בהמשך. החלק מורכב ממספר שכבות טרנספורמר שמחוברים בטור. כל טרנספורמר מוסיף ומעשיר את ההבנה של הרשת. מרבית הפרמטרים הנלמדים נמצאים בחלק זה, וכמו כן השוואה בין מודלים נעשת לפי התצורה שלו. שכבות הטרנספורמר לא משנות את גודל קלט, שהוא 256 על 384.

החלק האחרון הוא הסקת הרשת. לאחר שהרשת המירה את האסימונים לייצוג שהיא מבינה ופענחה את הקלט, היא יכולה להסיק מסקנות. המודל נדרש ליצור התפלגות הסתברויות עבור האסימון הבא לפי ההקשר שהוזן. החלק מורכב משתי שכבות: שכבות נרמול ושכבת dense. שכבת הנרמול נועדה לוודא יציבות בלמידה, ואילו שכבת dense יוצרת את התפלגות ההסתברויות. לא נעשה שימוש של פונקציית softmax בשכבה האחרונה, ולכן פלט הרשת הוא ערכי logits עבור כל אסימון. הסיבה לכך היא שחיזוי logits מאפשר למדינה יותר יציבה עם פונקציית השגיאה cross entropy loss ואלגוריתם הסמפלר צריך את ערכי logits כדי להוסיף temperature.



איור 32: דיאגרמה המתארת את המודל הראשון שעליו בוצע האימון. המודל מורכב משכבות הטמעה, קידוד מיקום, שש יחידות טרנספורמר, שכבת נרמול ושכבת dense סופית.

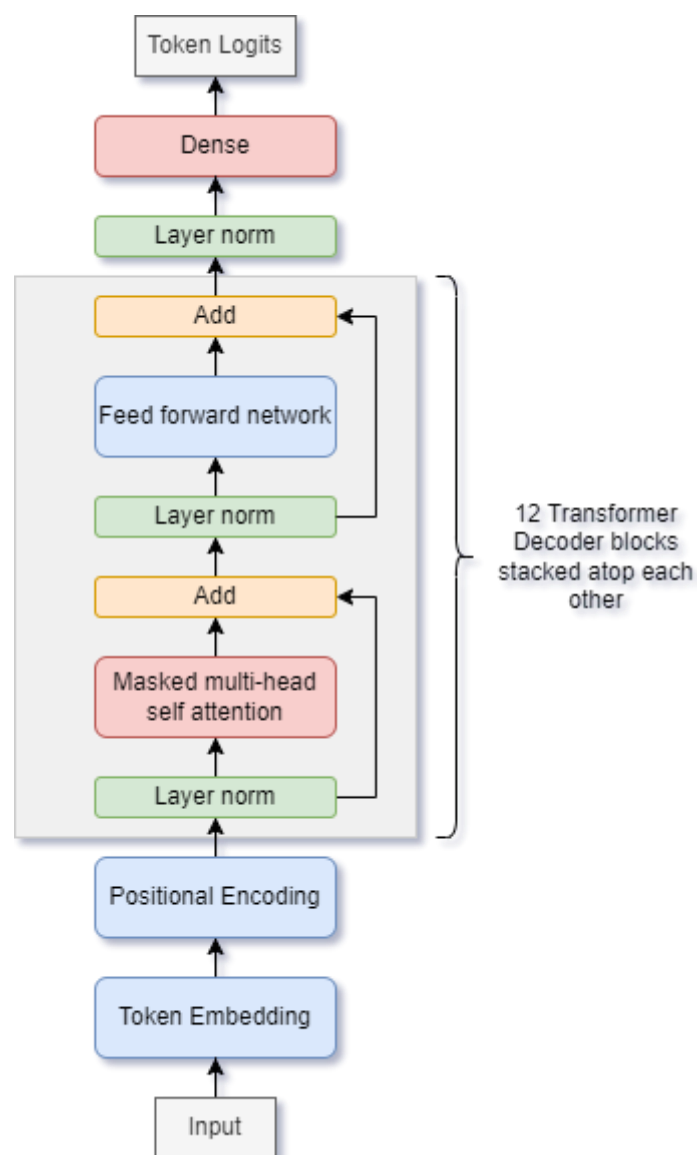
## המודל השני

כאמור המודל השני נלקח מפרויקט Open Pretrained Transformers שנועד לאפשר לאנשים גישה למודלים שפתיים שעברו את האמון הראשוני. כפי שתואר במבוא, תהליך האמון הראשון נורש מאגרי מידע אדירים ויכולות מחשוב מרחיקי לכת. כל מודל שפתי חייב לעבור את האימון הראשוני הזה כדי שיהיה בקיט בשפה. לכן חברת מטה סבסדה את אימונם של מודלים שונים מבוססי טרנספורמר ופרסמה את רובם לשימוש מחקרי.

המודל שנעשה בו שימוש במסגרת פרויקט זה הוא opt\_125m\_en, שגודלו כ-125 מיליון פרמטרים. המודל כמעט זהה למודל הראשון, עם כמה שינויים קטנים. אוצר המילים של tokenizer גדול הרבה יותר, 50,257 לעומת 2,500. החלק השני של המודל, חלק הפענוח שמכיל את יחידות הטרנספורמר, גדול הרבה יותר. ניתן לראות את ההבדלים בין תצורות המודלים בטבלה הבאה.

<i>Model</i>	<i>#L</i>	<i>#H</i>	<i>d<sub>model</sub></i>	<i>Seq_length</i>	<i>Vocab</i>	<i>Parameters</i>
article_10m	6	12	384	256	2,500	10,198,468
chatbot_125m	12	12	768	2048	50,257	125,237,760

טבלה 1: סיכום ההיפר-פרמטרים של המודלים שאומנו. טבלה זו מכילה את תצורת הארכיטקטורה של המודלים: מספר יחידות הטרנספורמר (*#L*), מספר ראשי התושמת לב (*#H*), גודל ממד ההטמעה (*d<sub>model</sub>*) ואורך טווח ההקשר (*Seq\_length*). בנוסף מוצג גודל אוצר המילים של tokenizer של כל מודל (*Vocab*).



איור 33: דיאגרמה המתארת את המודל שני שעליו בוצע האימון. המודל מורכב משכבות הטמעה, קידוד מיקום, שתיים עשרה יחידות טרנספורמר, שכבת נרמול ושכבת dense סופית.

### פונקציית המחיר

המודל שנבנה במסגרת פרויקט זה למעשה פותר בעיית סיווג. הוא צריך להחליט איזה מבין אלפי האסימונים האפשריים יתאים בצורה הטובה ביותר להקשר שקיבל. על כן נדרשת פונקציית שגיאה שערכה מייצג את המרחק בין התפלגות האסימונים שנתקבלו על ידי הרשת, לעומת האסימון המתאים

במאגר המידע. ישנם מספר פונקציות שיכולות להתאים לבעיה זו. לבסוף נבחרה פונקציית השגיאה cross entropy loss המכונה גם log loss.



איור 34: סרטוט של הפונקציות  $-\ln(x)$  באדום ו- $-\ln(1-x)$  בכחול בתחום  $0 \leq x \leq 1$ .

הפונקציה מבוססת על התכונות של פונקציות לוגריתמיות. אם נתבונן בגרף הפונקציה  $-\ln(p)$  נוכל לראות כי עבור  $p=1$  ערך הפונקציה הינו 0. לעומת זאת, כאשר הערך של  $p$  קטן ואף שואף ל-0, ערך הפונקציה גדל. אלה התכונות שאנו מחפשים בפונקציית השגיאה עבור חיזוי ההסתברות של האסימון הנכון. ככל שההסתברות קרבה ל-1 ערך השגיאה תקטן, ואילו עבור ערכי הסתברות קטנים, ערך הפונקציה תגדל.

פונקציית השגיאה  $-\ln(p)$  נכונה רק עבור ההסתברות של האסימון הנכון. אולם, לא מספיק שהפונקציה תיתן דיוק גבוהה למציאת האסימון הנכון, אלא גם למתן הסברות קטנה לשאר האסימונים. אם נשתמש בפונקציה  $-\ln(1-p)$  עבור אסימונים לא נכונים נקבל פונקציית שגיאה שעומדת בדרישות. כאשר ההסתברות הינה 0, ערך השגיאה גם הוא 0. ואילו ערכי הסתברות גדולים מובילים לשגיאה גדולה.

$$\text{loss}(\hat{y}, y_{\text{real}}) = \begin{cases} -\ln(\hat{y}), & y_{\text{real}} = 1 \\ -\ln(1 - \hat{y}), & y_{\text{real}} = 0 \end{cases}$$

נוסחה 16: פונקציית השגיאה כאשר היא מחולקת לפי ההסתברות האמיתית של האסימון,  $y_{\text{real}}$ .

מאחר וערך ההסתברות האמיתית  $y_{\text{real}}$  חייב להיות שווה ל-0 או 1, ניתן לרשום את הנוסחה בשורה אחת.

$$\text{Loss}(\hat{y}, y_{\text{real}}) = -y_{\text{real}} \times \ln(\hat{y}) - (1 - y_{\text{real}}) \times \ln(1 - \hat{y})$$

נוסחה 17: נוסחת פונקציית השגיאה בשורה אחת.

כדי לקבל את פונקציית החמיר, נעשה ממוצע לערכי השגיאה של כל האסימונים.

$$\text{Cost}(\hat{y}, y_{\text{real}}) = -\frac{1}{n} \sum_{i=1}^n y_{\text{real}_i} \times \ln(\hat{y}_i) + (1 - y_{\text{real}_i}) \times \ln(1 - \hat{y}_i)$$

נוסחה 18: הייצוג המתמטי המלא של פונקציית המחיר  $\text{binary cross entropy}$ .

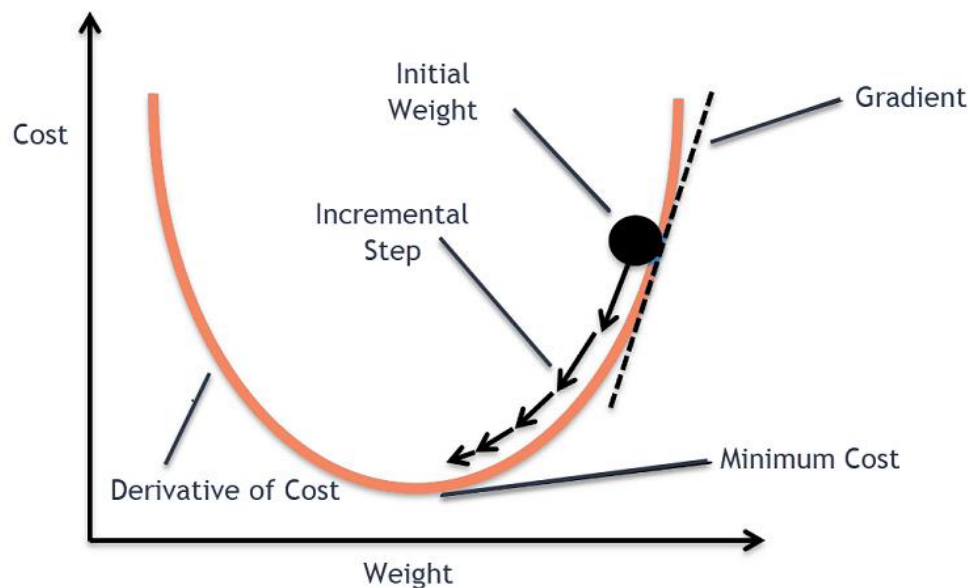
עם זאת, נעשה הנוסחה שפותחה היא עבור  $\text{binary cross entropy loss}$ , אולם בפרויקט נעשה שימוש בפונקציית המחיר  $\text{categorical cross entropy loss}$ . הן מבוססות על אותן עקרונות ובעלות אותן תכונות, אך השנייה מתאימה למקרים בהן יש מספר גדול של קטגוריות.

$$\text{Cost}(\hat{y}, y_{\text{real}}) = -\frac{1}{n} \sum_{i=1}^n \sum_{c=1}^c y_{ic} \times \log(\hat{y}_{ic})$$

נוסחה 19: נוסחת פונקציית המחיר  $\text{categorical cross entropy loss}$ . הקבוע  $c$  הוא מספר הקטגוריות, במקרה זה מספר האסימונים של המודל.

## אלגוריתם ההתכנסות (optimizer)

אלגוריתם התכנסות אחראי על עדכון הפרמטרים של הרשת לפי פונקציית המחיר בצורה שתוביל להפחתת המחיר ושיפור תוצאות המודל. optimizer הבסיסי ביותר הוא  $\text{gradient descent}$ . האלגוריתם מחשב את המחיר של כל הדגמים בepoch אחד, ובתהליך החלחול לאחר משתמש בערך הנגזרת החלקית של ערך המחיר ביחס לכל פרמטר כדי לעדכן אותם. כאשר הנגזרת שלילית, הגדלת הפרמטר יוביל להקטנת ערך המחיר. לעומת זאת, כאשר הנגזרת חיובית, הקטנת הפרמטר יגרום להקטנת ערך המחיר.



איור 35: סקיצה של פונקציית מחיר. ניתן לראות שכאשר הנגזרת של פרמטר ביחס לפונקציית המחיר חיובי, הקטנתו תוביל לערך מחיר קטן יותר. לעומת זאת, כאשר הנגזרת שלילית, הגדלת הפרמטר תוביל לערך פונקציית מחיר קטן יותר.

המתכנת יכול לשלוט בקצב העדכון באמצעות משתנה מכונה learning rate, שקובע בכמה לשנות את הפרמטרים בכול עדכון. ערך קטן מדי יוביל להתכנסות איטית, ואילו ערך גדול מדי עלול לגרום לאי יציבות ואף ימנע התכנסות. בשנים שחלפו מאז שהגו את האלגורית, התווספו שינויים שמשפרים את מהירותו ותוצאותיו.

$$w_{new} = w_{old} - lr \times \frac{\partial Cost(w_{old})}{\partial w_{old}}$$

נוסחה 20: נוסחת עדכון ערך פרמטר לפי האלגוריתם gradient decent. הפרמטר החדש שווה לפרמטר הישן פחות הנגזרת החלקית של פונקציית המחיר לפי הפרמטר כפול קצב הלמידה.

אחד השינויים המשמעותיים היא חלוקת מאגר הנתונים לקבוצות אימון. הרעיון מאחורי stochastic gradient descent הוא לעדכן את הרשת עבור כל פריט במאגר הנתונים, ולא פעם אחת עבור כלל הנתונים. הנגזרת של דוגמה אחת אמנם לא מדויקת עבור שאר הנתונים במאגר, אולם תדירות העדכון מאפשרת התכנסות מהירה מאוד, במיוחד למאגרי מידע גדולים. בנוסף, האקריות של העדכונים מונע התכנסות לנקודות מינימום מקומיות ועוזרת למודל להתכנס למינימום מוחלט. מרביתם לחלק את מאגר הנתונים למספר קבוצות המכונות batches, ועבור כל קבוצה עושים חלחול קדימה ואחורה. כך אפשר לנצל את מהירות ההתכנסות של stochastic gradient descent, ולשמור על ערך נגזרת מדויק.

לצורך אימון המודלים נבחר האלגוריתם Adam בזכות היעילות שלו. optimizer משתמש בקצב למידה שונה ומותאם עובר כל פרמטר, דבר אשר משפר את תוצאותיו ויעילותו. יתר על כן, האלגוריתם

עושה שימוש בתאוצת שינוי הפרמטרים בנוסף לנגזרת הbatch הנוכחי כדי לעדכן את הרשת. ניתן לחשוב על התאוצה כזיכרון optimizer מקבוצות האימון הקודמות. אילו הנגזרת של פרמטר הייתה חיובית עבור כל קבוצות האימון מלבד אחת, התאוצה תהיה חיובית. כאשר האלגוריתם יעדכן את הפרמטר לפי הנגזרת השלילית, הוא יתחשב בתאוצה ובכך יפחית את השפעת הנגזרת היוצאת דופן.

## תהליך ותוצאות האימונים

### אימון המודל הראשון

אימון המודל הראשון נערך במחשב האישי שלי, בעל כרטיס מסך מסוג RTX 2060 עם 6GB של זיכרון. הסיבה לכך היא שהאימון הראשון לוקח זמן רב ומאגר מידע אדיר. שימוש בשירותי מחשוב ענן, כדוגמת Google Colab, היו עולים סכומי עתק. בנוסף, רציתי ללמוד כיצד מאמנים מודל באמצעות כרטיס מסך ולא עם מעבד. עבור סביבת tensorflow במערכת ההפעלה windows הדבר יוצר אתגר לא פשוט.

לסביבת tensorflow אין גישה ישירה לכרטיס המסך. צריך להוריד סביבת הרצה בשם anaconda וליצור יישום בעל cuda toolkit עם ההגדרות הנכונות. הבעיות לא נגמרות כאן. ספריית tensorflow סיים את תמיכתו במערכת ההפעלה עבור גרסאות 2.11 ומעלה. הרצתי מספר מודלים קטנים בסביבה זו כדי לבדוק את ביצועיהם, אולם האימון של המודל הסופי נעשה בסביבת לינוקס. מערכת ההפעלה תומכת גם בגרסאות החדשות של tensorflow.

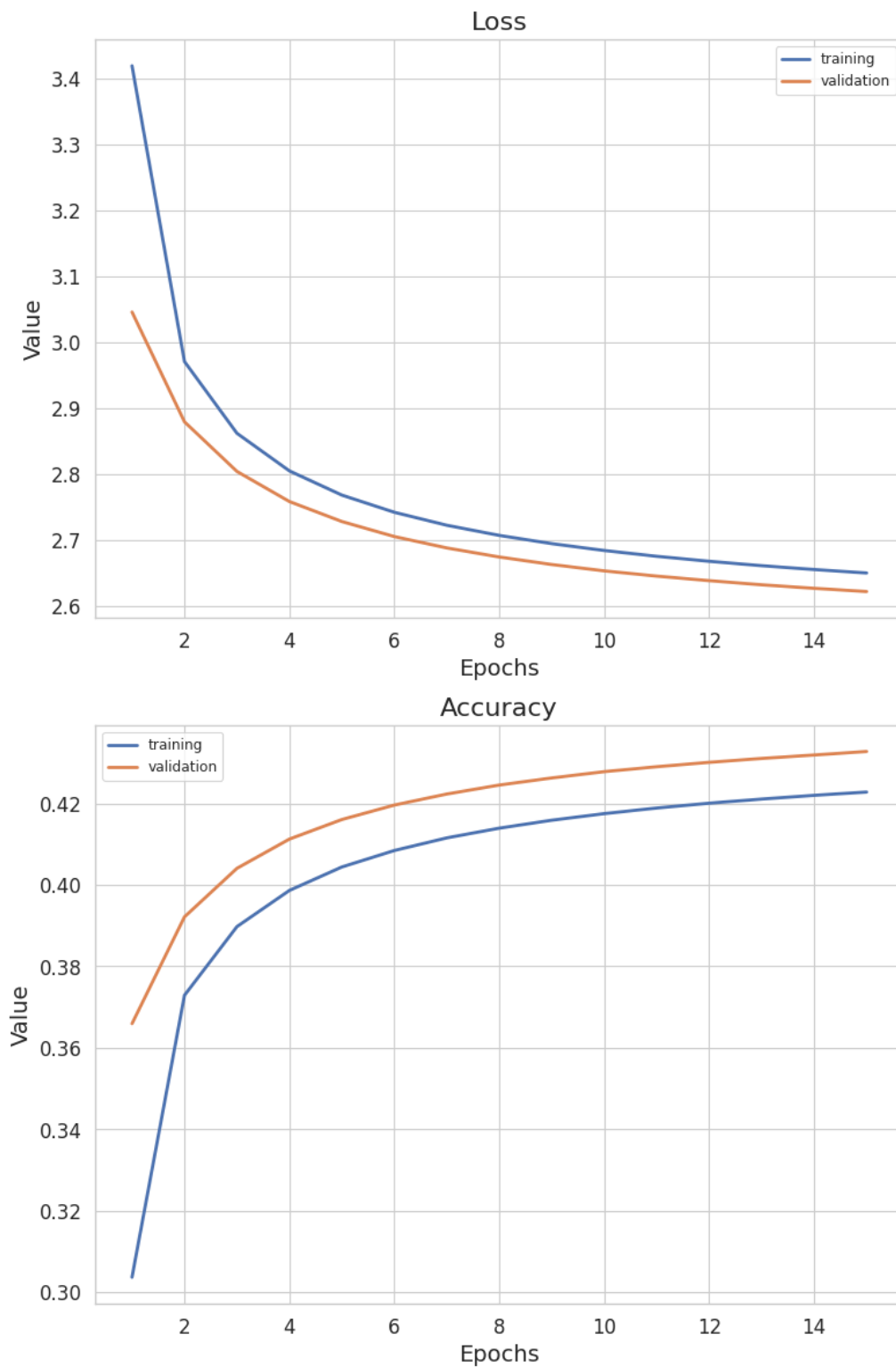
לאחר ניסוי וטעיה מצאתי שהכרטיס מסך יכול לאמן את המודל בbatches שגודלם 48. ככל שמספר האיברים בקצובה גודל יותר, דיוק החלחול לאחר טוב יותר. בנוסף, קבוצות גדולות יותר ככלל מובילות לאימון מהיר יותר. קצב הלמידה נבחר בעיון במאמרים המתארים אימון של מודלים מבוססי טרנספורמרים אחרים, הערכתי שערך מתאים לגודל המודל הוא  $10^{-4}$ .

תהליך האימון המלאה התחלק לשני חלקים. בחלק הראשון המודל התאמן על 10 epochs כאשר לכל epoch לקח כשעתיים ו36 דקות. לאחר יום שלחתי את המודל להתאמן לעוד 5 epochs עם אותם ההגדרות. בזכות גודל מאגר המידע לא נצפו בעיות של overfitting. המודל ראה כל פריט 15 פעמים בלבד. השימוש בשכבות dropout לאורך חלקי המודל מונע עוד יותר את הסיכון לoverfitting.

Model	Learning rate	Epochs	Batch size	Training steps
article_10m	$10^{-4}$	15	48	407,400

טבלה 2: סיכום ההיפר-פרמטרים של אימון המודל. הטבלה מציגה את קצב הלמידה, מספר הepochs, גודל קבוצות האימון ומספר צעדי האימון הכולל.





איור 36: סרטוטים של גרפים המתארים את ביצועי המודל הראשון בזמן האימון.

כדי לאמוד את ביצועיהם של מודלים שפתיים יצרנים נהוג להשתמש בשלושה ערכים: הפסד, דיוק perplexity והפסד הוא ערך פונקציית המחיר עבור כל מאגר הניסוי, כאשר מדובר ב-cross entropy loss הוא מסמל את ההבדל בין קלט המודל לבין הפלט שלו. הדיוק מייצג את אחוז החיזויים הנכונים מתוך כלל חיזוי המודל. לבסוף ערך perplexity מייצג את בלבול המודל, כלומר ההפך מביטחון גבוהה בתשובותיו נכונות. כאשר הערך קטן, המודל יותר בטוח בתשובות שלו.

$$Perplexity = e^{Loss}$$

נוסחה 21: הנוסחה לערך Perplexity של מודל. יש לציין כי הבסיס של הפונקציה המעריכית תלוי בבסיס של הפונקציות הלוגריתמיות cross entropy loss. בסביבת tensorflow ערך זה הוא e, אולם בסביבות אחרות הוא עשוי להיות 2.

Model	Loss	Accuracy	Perplexity
article_10m	2.6291	0.4294	13.8613

טבלה 3: טבלה המרכזת את תוצאות האימון של המודל: הפסד דיוק perplexity.

בגלל גודל המודל והזמן שנדרש לאמן אותו, לא ניתן לשנות את ההיפרפרמטרים הרגילים כדי לבדוק את השפעתם על תוצאות המודל. עם זאת, ניתן לערוך סדרת ניסויים כדי למצוא את ערכי הפרמטרים הטובים ביותר עבור הספלר למטרות שונות: top\_pi temperature. כזכור ערכים אלה מאפשרים למשתמש לשוט על אקריות וביטחון העצמי של המודל בעת בחירת האסימונים. ערכים שונים מתאימים למשימות שונות.

אוסף פלטים של המודל לאחר אימון נצא בנספחים (עמוד 169). כדי ליצור את הכתבות, נתתי למודל כקלט משפט אחד שמציג את הנושא הכללי של התכבה ומשמש כפתיח. שאר תוכן הכתבה נוצר על ידי המודל בלבד. לאחר תהליך של ניסוי וטעייה מצתי כי הגדרות סמפלר סבירות הן temp=1 וtop\_p=0.5i. ייתכן כי מחקר יותר מעמיק של תצורות שונות היה מוביל לפלטים יותר טובים. לא מצאתי כלי שיוכל לדרג את הכתבות באופן אוטומטי, ולכן מחקר זה היה לוקח כמות זמן מופרזת.

ניתן לראות כי המודל למד לאיית מילים, ולא נמצאו שגיאות כתיב חמורות כלל. מרבית המילים ניתנים להצגה על ידי אסימון אחד או שניים, אולם גם מילים המורכבות ממספר גדול של אסימונים נכתבו נכון, לדוגמה responsibility, infrastructure, earthquake, unemployment referendum ועוד. הוא גם למד להשתמש במילים בצורות הנכונות ותמורות הנכונות. בנוסף המודל הפגין בקיאות בביטויים שונים ובטון המילים.

ברמת משפטים שלמים, המודל מרבה להצליח להרכיב משפטים תקינים. המבנים התחביריים נכונים וכוללים נושא ונשוא ברורים. המודל מוסיף פיסוק הגיוני ומפגין הבנה בציטוטים, אם כי הוא מרבה לשים מילים בגרשיים גם כשאין בכך צורך. ניכרת בעיה שהמודל לעיתים מוסיף רווח לפני נקודה. לאחר

בדיקה עולה כי הבעיה קיימת גם במאגר הנתונים, ולכן אין זה מצביע על למידה לא נכונה של המודל. עוד בעיה היא השימוש המיותר בתמורות. המודל נוטה לחזור על דבריו באמצעות תמורות שלא מוסיפות תוכן ולא מקלות על הקורא.

המודל סובל ברמת הטקסט. הוא אמנם כותב משפטים תקינים, אולם אין ביניהם קשר הגיוני. לא עומד מאחורי הכתבות חוט מחשבה וניכר שלא מדובר בנושאים או אירועים קונקרטיים. המודל כן מפגין בקיאות בקשרים בין מושגים, למשל אם מדובר בישראל הוא יכתוב על המזרח התיכון, על בנימין נתניהו, על הפלסטינים, על איראן וכדומה. אך הרעיונות לעיתים קרובות סותרים. יתכן והמודל יכתוב על מתקפה של איראן נגד פקיסטן, ואז יוסיף שפקיסטן השיבה במתקפת נגד בעיראק.

לבסוף, המודל מצליח לדבוק באותו נושא. היה מצופה שאחרי כמה משפטים הוא יקנה למשפט הראשונה חשיבות הולכת וקטנה, ותוך זמן קצר יתחיל לכתוב על נושאים אחרים. העובדה שהדבר לא קורה מצביע על עוצמתו של מנגנון התשומת לב ולכך שהמשפטים שכן נוצרו על ידי עוסקים באותו נושא. יש להזכיר שמדובר במודל מאוד קטן ביחס למודלים אחרים בתחום עיבוד השפה הטבעית, כדוגמת GPT 3 עם 175 מיליארד פרמטרים. אני מאמין שהמודל מפגין יכולות שפתיות מרשימות מאוד וכי מודל מעט גדול יותר היה מצליח לכתוב כתבות קוהרנטיות.

### אימון המודל השני

האימון של המודל השני נעשה באמצעות שירותי מחשוב ענן של Google Colab בגלל הגודל שלו. כזכור למודל יש כ-125 מיליון פרמטרים, כמות שלא ניתן להריץ על כרטיס המסך שברשותי. בנוסף, האימון הוא מין fine tuning, ולכן ניתן לערוך אותו במספר קטן של epochs. אימון המודל נעשה בשלושה חלקים עקב מדיניות השימוש בכרטיס המסך T4 המוגבל לשעה וחצי של שימוש רציף ביממה.

לאחר ניסוי וטעיה נתגלה כי כרטיס המסך T4 בעל 15GB של זיכרון יכול לאמן את המודל עם גודל batch של 20. בדומה למודל הראשון רצוי לאמן את המודל עם קבוצות אימון גדולות ככל האפשר. מאחר ומותר לאמן את המודל למשך כשעה וחצי, האימון התבצע בחלקים של 2 epochs. בחלק השישי, לאחר epoch הראשון, ראיתי כי ערך ההפסד של המודל קטן מההפסד של מדגם המוחן. תופעה זו מצביע על overfitting, ולכן החלטתי לעצור את האימון.

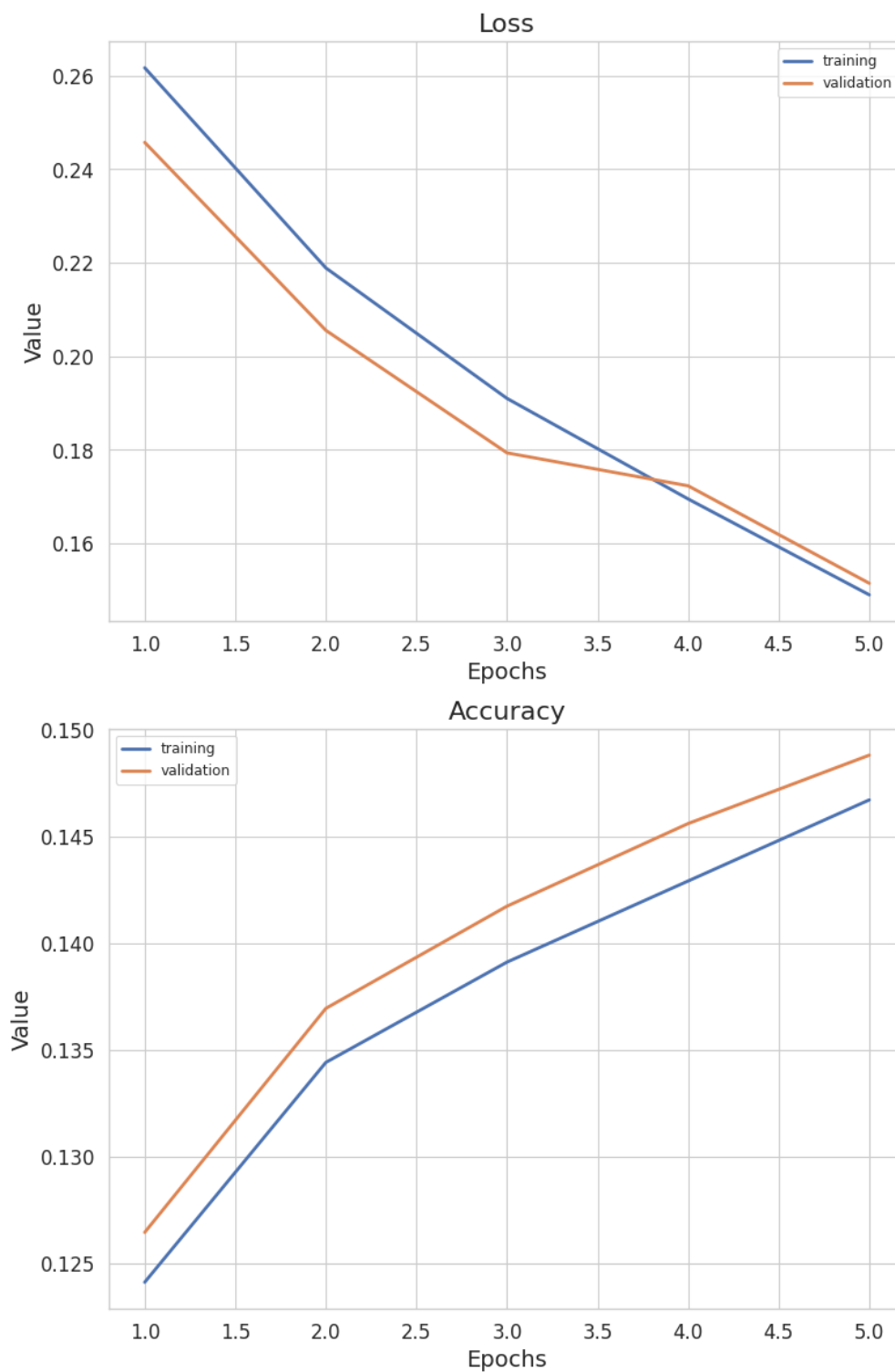
נשאלת השאלה מדוע המודל מראה סימני overfitting לאחר רק חמישה epochs. בעיניי למודל יש שתי דרכים להקטין את פונקציית המחר, ובכך להשתפר. הוא יכול לעבד את המידע, לעשות אבסטרקציה ולהשיב עם התשובה המדויקת ביותר. דרך זו מדייקת את התוכן של התשובה, שלעיתים יכול להיות אסימון אחד, למשל כאשר שואלים מהי בירת ישראל. ההבדל בין תשובה נכונה לבין תשובה שגויה היא מילה אחת.

לעומת זאת, המודל יכול להשתפר בתבנית השאלה, כלומר במסגרת של התוכן. בדוגמה הקודמת המסגרת של התוכן היא: The capital of Israel is \_\_\_\_\_. ניתן להבחין בכך שהמסגרת מורכבת ממספר גדול של מילים ביחס לתוכן עצמו. על כן, אני סבור שהמודל התחיל לדייק את תבנית התשובה יותר מאשר את התוכן שלה. המודל למד את תבניתם של התשובות שהיו במאגר הנתונים לאימון ולא הופיעו במאגר הבוחן. ניתן לראות תופעה זו גם אצל מודלים גדולים כדוגמת ChatGPT. תשובות המודל תמיד יהיו מנוסחות ומוצגות בצורה משכנעת, אולם לעיתים קרובות התוכן עצמו כזבי לחלוטין. תופעה זו מתרחשת לעיתים כה קרובות שהיא זכתה לכינוי משלה, hallucinations.

כזכור מההסבר על מאגר המידע, נעשה שימוש ב-sample\_weights אשר קובעים את משקלם של איברי הקלט על ערך פונקציית המחיר. השאלות עצמן נמצאות בתבנית כדי שהמודל יוכל ללמוד היכן נמצאת השאלה ולהסביר לו את משימתו. התבנית הזו קבועה, ולכן אין לתת לה משקל שווה לזה של התשובה עצמה.

Model	Learning rate	Epochs	Batch size	Training steps
chatbot_125m	$10^{-4}$	5	20	10,770

טבלה 4: סיכום ההיפר-פרמטרים של אימון המודל. הטבלה מציגה את קצב הלמידה, מספר הepochs, גודל קבוצות האימון ומספר צעדי האימון הכולל.



איור 37: ביצועי המודל השיחתי בזמן האימון. הערכים הללו מושפעים על ידי ערכי ה-*sample weights* שצוינו בהסבר על יצירת מאגרי המידע. לכן, לא ניתן להשוות את מדדי הביצועי הללו למודלים אחרים.

כדי לאמוד את ביצועיהם של מודלים שפתיים יצרנים נהוג להשתמש בשלושה ערכים: הפסד, דיוק ו-perplexity. ההפסד הוא ערך פונקציית המחיר עבור כל מאגר הניסוי, עובר cross entropy loss הוא מסמל את ההבדל בין קלט המודל לבין הפלט שלו. הדיוק מייצג את אחוז החיזויים הנכונים מתוך כלל חיזויי המודל. לבסוף ערך perplexity מייצג את בלבול המודל, כלומר ההפך מביטחון גבוהה בתשובותיו נכונות. כאשר הערך קטן, המודל יותר בטוח בתשובות שלו.

Model	Loss	Accuracy	Perplexity
chatbot_125m	0.1574	0.1472	1.1705

טבלה 5: טבלה המרכזת את תוצאות האימון של המודל השיחתי: הפסד, דיוק ו-perplexity.

אוסף פלטים של מודל לאחר האימון נמצא בפרק הנפסחים (עמוד 178). נאספו שלאות כדי לבחון את המודל במספר תחומים: הבנתו את סוגיות הקיימות, בקיאותו בידע כללי ויכולות הבנת ההוראות שלו. בניגוד למודל הראשון, המודל השיחתי יכול ליצור מגוון של תכנים. לכן, הגדרות סמפלר שונות יובילו לתשובות טובות יותר עבור נושאים שונים. עם זאת, החלטתי לא ליצור מספר תשובות לאותה שאלה עם הגדרות סמפלר שונות ולבחור את הטובה ביותר. לדעתי, הדבר אינו שונה מליצור תשובות עד שבמקרה תצא תשובה טובה, כלומר לשחק עם התוצאות.

בניתוח התוצאות ניתן לראות כי המודל מפגין שליטה בלתי רגילה בשפה. הוא כותב משפטים שלמים עם פיסוק נכון ומבנים תחביריים נכונים שמתאימים להקשר. בחירת המילים שלו התאימה לטון של כל תשובה ואורך התשובות תואם לשאלות. הוא ככלל לא נוטה לחזור על אותן המילים ומשפטים ובניגוד למודל הראשון אינו מוסיף תמורות מיותרות. אולם, בשאלה מספר 9 "What are the benefits of using the Python programming language", הוא חוזר על אותו יתרון לאורך כל התשובה. הדבר נובע מהעובדה שמאגר הנתונים כלל כמות קטנה מאוד של מידע אודות שפת התכנות פיתון. אוסיף ואציין שגם כאן המודל בחר באורך תשובה ראוי.

המודל מפגין הבנה עמוקה בתחום הקיימות ומצליח לענות על שאלות מגוונות. הוא כתב מספר הצעות לדרכים למנוע התחממות גלובלית. בנוסף, הוא מצליח לנסח את היתרונות בשיטות מחייה שונות ויכול להסביר את הגורמים להתחממות גלובלית. תוצאות המודל בסוגיית הקיימות עלו על צפיותי. יש לציין כי ייתכן והמודל סבל מoverfitting לשאלות העוסקות בקיימות, ולכן תשובותיו הן במידה רבה שינון מאגר האימון.

שליטתו של המודל בידע כללי היא חלקית. הוא אכן מצליח לענות על שאלות כגון מה היא בירת ישראל ומי כתב את הספר "To Kill a Mockingbird". אולם, כאשר נתבקש לציין שלושה אמנים מתקופת

הרנסנס הוא צדק רק באחד, אם כי שאר האנשים אכן היו אמנים. בנוסף, כאשר נשאל לגבי הנהר האורך ביותר בעולם, המודל ענה נכון, אך הוסיף שאורך הנהר הוא 4,000 מיל, דבר שאינו נכון. כאשר נשאל לגבי מזכ"ל ארצות הברית, המודל לא ידע לענות. היות המודל רשת בינה מלאכותית, הוא אינו יכולה לעשות תרגילי חשבון, ולכן כאשר נשאל לגבי תוצאת הביטוי  $1+1$  הוא השיב שהתוצאה היא 1.

יכולות הכתיבה החופשית של המודל מרשימות מאוד. כאשר הוא נתבקש לכתוב סיפור קצר על אדם וכלב, הוא יצר סיפור בעל מספר עקרונות ספרותיות. הוא כתב פתיחה שמציגה את הדמויות והסביבה של הסיפור. לאורך האירועים הוא שומר על סדר כרונולוגי הגיוני ובוחר לנקוב במקומות מתאימים. מרשים מכל, המודל יצר בעיה באמצע הסיפור והציג את הפתרון בסופו. האירועים אמנם לא נכתבו בצורה ספרותית, אולם הבנת המודל את הצורך בבעיה ופתרון מרשים מאוד בעיניי.

בתשובות במודל ניכר שיש חוט מחשבה מנחה. נדמה כאילו המודל אמר לעצמו מראש מה צריך להיות תוכן התשובה, אורכה וטון הדיבור שלה. אזכיר שוב כי המודל כותב את התשובות הללו מילה-מילה ואין לא יכולת לתכנן מראש. אני מאמין שמודל יותר גדול היה מצליח לזכור יותר עובדות. אולם, יש לציין שתהליך האימון היה עלול לדרוס ידע קודם לשם הבנת מבנה השאלה תושבה.

## שלב היישום

### הפשטת היישום

מטרת היישום היא לדמות את המערכות הנדרשות כדי לאפשר שימוש של המודלים על ידי הקהל הרחב, בדומה לחברות גדולות כדוגמת OpenAI, Meta, Microsoft וכדומה. היישום מבוסס על ממשק אינטרנטי שמפשט את פעולת המודלים מאחורי הקלעים. האתר נגיש לכל אדם בעל חיבור אינטרנט.

כדי לתפעל את המודלים צריך שני חלקים מרכזיים: שרת מחשוב ומנהל בקשות. שרת המחשוב מרכז מספר מחשבים שמריצים את המודלים ומאפשר שליחת בקשות מחשוב. הוא אחראי על הקמת הקשר בינו לבין ישויות המחשוב ולהתעדכן לגבי מצבן. מנהל הבקשות מתווך בין שרת האינטרנט לבין שרת המחשוב. הוא עוקב אחר הבקשות, מנהל עומס, מתאם בין סוגי הבקשות לסוגי השרתים ושולח את הבקשות לשרת המחשוב.

כאשר משתמש מפעיל את אחד המודלים, אתר האינטרנט יוצר בקשת מחשוב עם הפרטים המתאימים. הבקשה עוברת למנהל הבקשות שדואג לה. הוא ממין את הבקשות לפי העדיפות שלהן, מתקשר עם שרת המחשוב ומבקש משרת המחשוב הפניה לישות מחשוב פנויה שמתאימה למשימה. שרת המחשוב שולח לישות המחשוב את הבקשה וממתין לתשובה. כאשר מתקבלת תשובה, שרת המחשוב שולח אותה למנהל הבקשות שמעביר אותה לשרת האינטרנט.

כל חלקי היישום נכתבו בשפת פיתון. שרת האינטרנט, שרת המחשוב ומנהל הבקשות רצים באותה תוכנה לשם פשטות. חלקי התכונה ממודרים היטב, ולכן ניתן להעביר את החלקים לשרתים שונים במידת הצורך. ישויות המחשוב נמצאות באותה רשת פרטית עם שרת המחשוב, אולם אפשר להוסיף תמיכה לישויות מחשוב מחוץ לרשת.

### שרת אינטרנט

שרת האינטרנט מבוסס על ספריית flask בפיתון ועושה שימוש בטכנולוגיית websocket כדי לתקשר באופן יעיל עם המשתמשים. השרת מפעיל אתר שנגיש לאינטרנט באמצעות port forwarding. מאחר והוא רץ על הרשת הביתית שלי, יש חשיבות גדולה לאימות המשתמש לפני שהוא יוכל לגשת לדפי האתר השונים. על כן, לפני שהמשתמש מתקשר עם חלקי האתר השונים הוא מתבקש לאמת את זהותו באמצעות הזנת פרטי התחברות.

האימות נעשה באמצעות ספריית flask\_login שמנהלת את sessions של המשתמש ומפשטת את תהליך האימות. בנוסף נעשה שימוש בספריית flask\_sqlalchemy המאפשרת שימוש בבסיס נתונים. בכל בקשות העומדים אימות המשתמש נבדק, ובמידה ואינו מאומת הוא מופנה לדף האימות. בקשות



אחרות, בנוסף לניסיוני הקמת websocket, נחסמות כאשר המשתמש אינו מאומת. כך האתר מוגן משימוש לא מאושר ואף זדוני.

מלבד דף האימות, קיימים באתר עוד ארבעה עמודי תוכן. כל עמוד מבוסס על טכנולוגית templates של flask. הם יורשים מתבנית הבסיס שמכונה base.html ומכילה אזור ניווט בין דפי האתר ואזור תוכן של העמוד הנוכחי. אין למשתמש גישה לתבנית הזו אלא דרך ירושה של עמודים אחרים. האתר מכיל דף הסבר על הפרויקט, ממשק לשימוש במודל שיחתי, ממשק להפעלת המודל הכתובת ואזור המרכז את פרטי שרת המחשוב.

דפי ממשקי המודל מתקשרים עם האתר באמצעות טכנולוגית websockets. במודל האינטרנט של שרת לקוח רגיל, הלקוח מבקש מידע והשרת משיב. עם זאת, השרת לא יכול מיוזמתו לשלוח ללקוח מידע, הוא יכול רק להשיב לבקשות. המודל הזה עבוד עבור מודלים שמחזירים תשובות מהר, למשל מודל המסווג תמונה למספר קבוצות. אולם, במודל יצירתי התשובה נבנית אסימון-אסימון, תהליך שיכול לקחת מספר שניות עד דקות. בעוד המשתמשים ממתין לתשובה שבעיניו לא מגיע.

הפתרון הוא לשלוח למשתמש את חלקי התשובה, האסימונים. כך לנגד עיניו הוא יראה כיצד המודל מרכיב את התשובה שלו. המודל הבסיסי לא תומך בתקשורת כזו. פתרון אחד הוא לשלוח בקשות get כל עוד השרת לא שלח תשובה שהוא סיים. השיטה הזו אמנם עובדת, אך לא נהוג להשתמש בה מכיוון שהיא מסורבלת ולא מאפשרת תקשורת יותר מורכבת. הפתרון שנמצא הוא websockets.

טכנולוגית websockets דומה לsockets רגילים בין מחשבים. היא פותחת מין צינור תקשורתי בין הלקוח לבין השרת שדרכו זורם מידע. המשתמש יכול לשלוח איזה מידע שהוא רוצה לשרת, והשרת יכול לשלוח מידע כאוות נפשו. כאשר המשתמש שולח בקשה למודל, השרת מחזיר האסימונים שמרכיבים את התשובה כאשר הם נוצרים. הלקוח קוראה את התעבורה ומציג למשתמש את המידע שמתקבל.

כאשר לקוח משתמש באחד המודלים, שרת האינטרנט יוצר בקשה ושולח אותה למנהל הבקשות. שרת האינטרנט לא עוסק באף חלק מהחישובים. המשימה היחידה שלו היא להעביר את המידע שמתקבל מהבקשה לצד הלקוח. נוצרת בקשה אחת כל פעם שהמשתמש מתקשר עם אחד המודלים, וכאשר התשובה הסתיימה הבקשה נמחקת.

## שרת המחשוב

שרת המחשוב הוא שרת ראשי המנהל ישויות מחשוב המחוברות אליו. התקשורת מתבצעת האמצעות tcp sockets שמקמים כאשר הישויות המחשוב יוצרות חיבור עם השרת הראשי. כל socket רץ

בthread שונה כדי לאפשר להם להתנהל במקביל בלי להפריע אחד לשני. שרת המחשוב יוצר מחלקה עבור כל ישות שבאמצעותה ניתן לשלוח בקשות מחשוב.

מאחר ויש מספר גדול של בקשות בין הישויות לשרת הראשי, נעשה שימוש במודל packets, המחלק את הבקשות לפקטות שונות. כדי לשלוח פקטה היא צריכה לעבור תהליך של serialization, שהופך את תוכן המחלקה למידע שניתן לשלוח. deserialization היא התהליך ההפוך, שבו מפענחים את המידע יוצרים את המחלקה מחדש. כל מחלקה מקבלת מספר מזהה שמסייע למערכת לעבוד עם החבילות השונות.

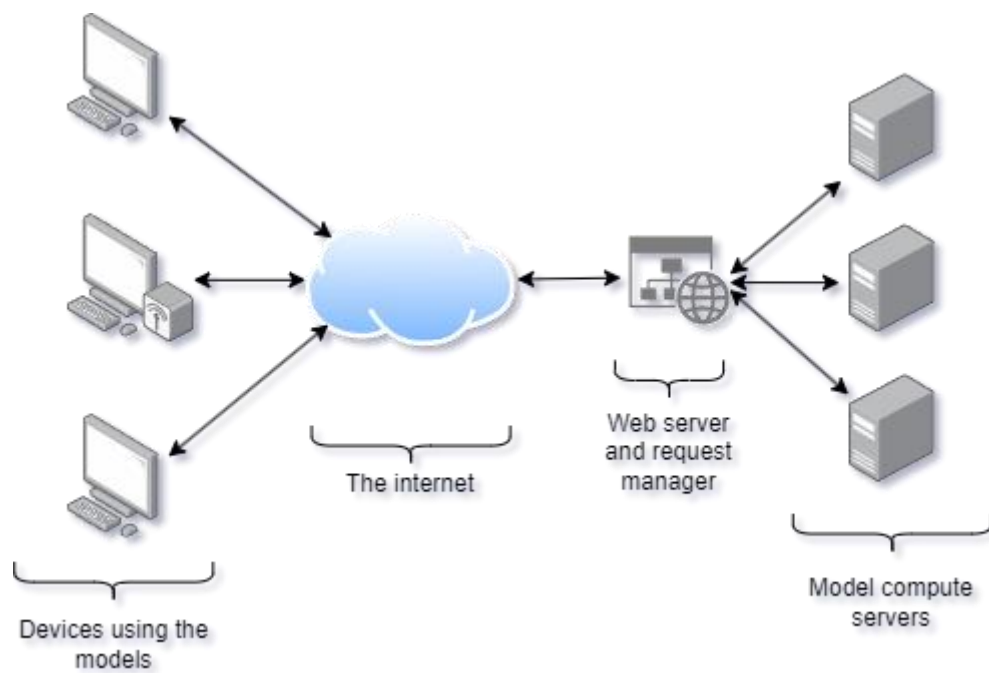
כדי לשלוח פקטה נוצר buffer של מידע המכיל בבית הראשון את המספר המזהה את הפקטה, ולאחר מכן את הייצוג הserialized שלה. כאשר הצד השני קורא את המידע שנשלח הוא קודם כל קורא את הבית הראשון כדי להבין באיזה פקטה מדובר. לאחר מכן הוא יכול לפנות למחלקת החבילה וליצור אותה מחדש באמצעות המידע שנשלח. הפועלות הללו נעשות עם פונקציות encode\_packet וdecode\_packet שמומשו בקובץ הפקטות. שליחת וקבלת המידע נעשים באמצעות הפונקציות של מחלקת socket עזמה.

יש פקטות עובר הפעולות הבאות:

1. שליחת בקשת מחשוב
2. קבלת תשובת מחשוב
3. שליחת מידע אודות שרת המחשוב

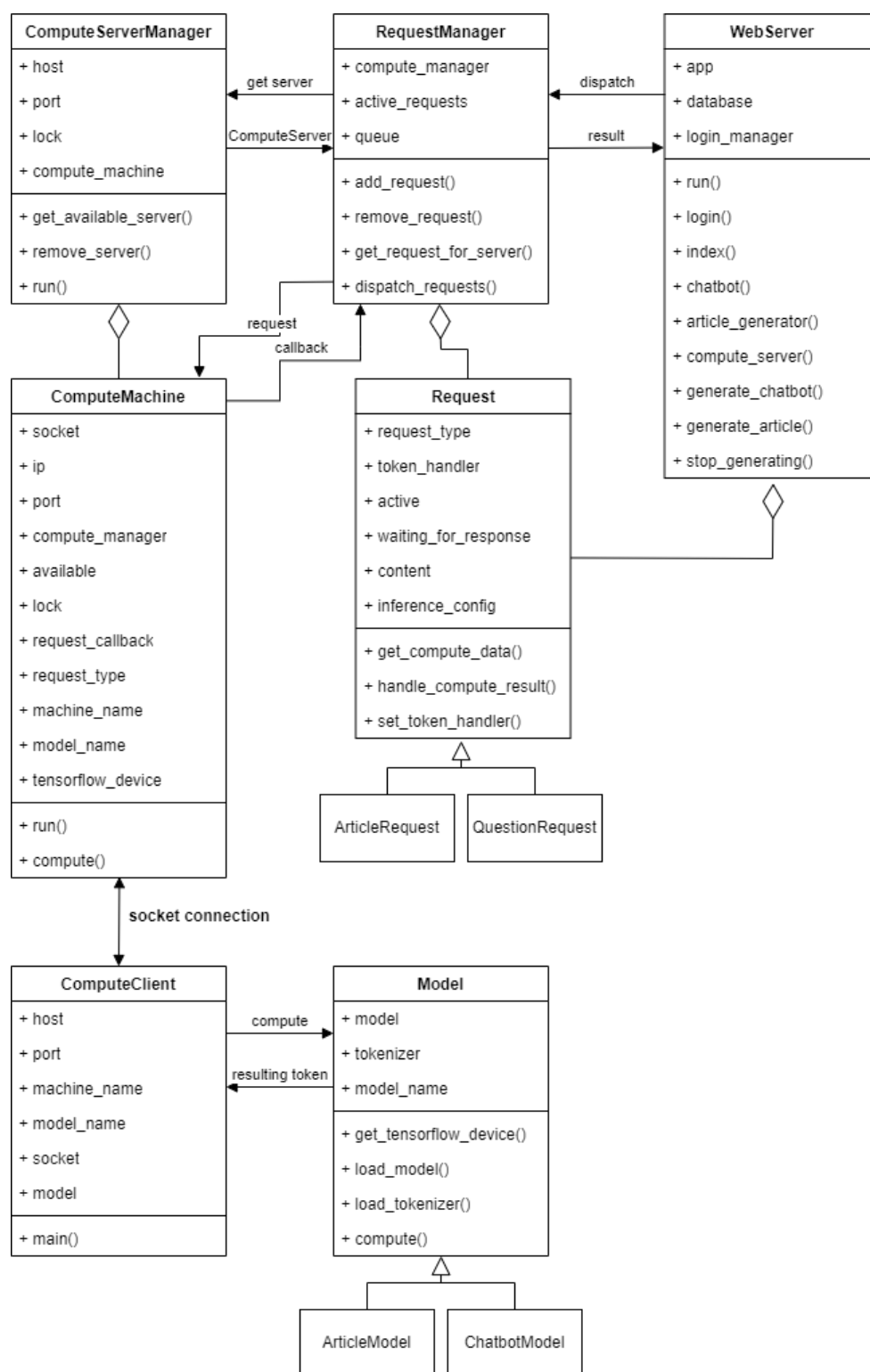
## מנהל בקשות

כדי לחבר בין שרת האינטרנט לבין שרת המחשוב צריך מתווך שינהל את בקשות המחשוב. את התפקיד ממלא מנהל הבקשות. כאשר שרת האינטרנט שולח למהל בקשה, הוא מוסיף אותה לרשימת הבקשות הפעילות. המנהל מתשאל את שרת המחשוב אילו שרתים פנויים ואילו מודלים הם מריצים. הוא עובר על רשימת הבקשות ומתאם בין בקשות לבין שרתים פנויים. בנוסף לכך, נעשה תהליך של תיעדוף הבקשות. ביישום התיעדוף מחולק באופן שווה, ועל כן הבקשות נעות בתור המתנה בצורה מעגלית.



איור 38: דיאגרמה המתארת את עבודת היישום. מכשירים המחוברים לאינטרנט מתקשרים עם האתר שמציג בפניהם ממשק לשימוש במודלים. כאשר משתמש מפעיל מודל, האתר מפנה את בקשות המחשוב לשרתי מחשוב המחוברים עליו, ומעביר את התוצאות למשתמש.

## תיאור UML של מחלקות היישום

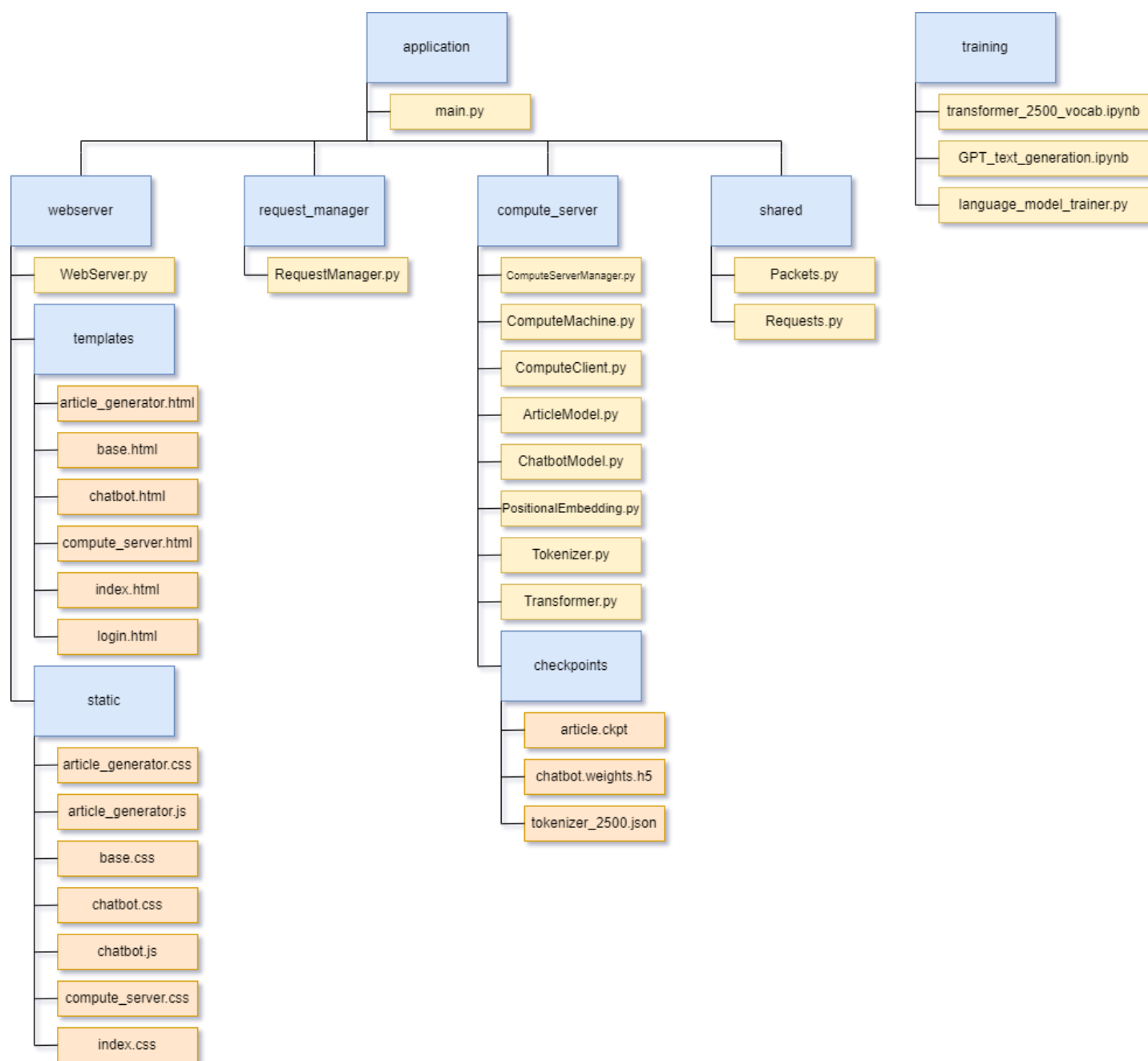


## מדריך למפתח

### מבנה תיקיית הפרויקט

הפרויקט נכתב שתי סביבות עבודה עיקריות: סביבת פיתוח בענן של Google Colab וסביבת פיתוח מקומית PyCharm. מבנה תיקיית הפרויקט וסוגי הקבצים משקפים עובדה זו. חשוב לציין כי חלק מהקוד שפותח בענן הורד לשם הרצה מקומית. אי לקח, יצורפו קבצים שמנחים את לוגיקת הרצת הקוד. חלקים קטנים מהפרויקט נכתבו ישירות בinterpreter של פיתון. נעשה מאמץ לצרף את קטעי הקוד הללו, אולם יתכן שחלקם התפספסו.

כלל התייעוד בקוד נכתב באנגלית. בנוסף, הערות שמתעדות את מטרתם של קטעי קוד ומחלקות נכתבו באנגלית בזמן הפיתוח. לכן, חלק מהתייעוד יופיע בעברית וחלק יופיע באנגלית.



איור 39: דיאגרמה המתארת את מבנה תיקיית הפרויקט. היא מורכבת משני עצים עקרים: אימון המודלים והיישום.

## תוכן עניינים של המדריך למפתח

66.....	מדריך למפתח
66.....	מבנה תיקיית הפרויקט
68.....	תוכן עניינים של המדריך למפתח
71.....	הכנת הנתונים, בניית ואימון המודל הראשון
71.....	קבועים
72.....	הכנת הנתונים
72.....	הורדת מאגר הנתונים
73.....	יצירת ואימון tokenizer
82.....	יצירת מאגר הנתונים לאימון
83.....	מנגנון התשומת לב
88.....	שכבת החלחול קדימה
89.....	יחידת הטרנספורמר
91.....	קידוד מיקום
93.....	קוד אימון המודל
97.....	הכנת הנתונים, בניית ואימון המודל שני
97.....	יצירת והכנת מאגר הנתונים
99.....	טעינת המודל ואימונו
101.....	יישום
101.....	main.py
102.....	שרת אינטרנט

102.....	WebServer.py
106.....	login.html
107.....	base.html
108.....	base.css
109.....	index.html
110.....	index.css
111.....	chatbot.html
112.....	chatbot.css
113.....	chatbot.js
116.....	article_generator.html
117.....	article_generator.css
118.....	article_generator.js
120.....	compute_server.html
121.....	compute_server.css
122.....	מנהל בקשות
122.....	Requests.py
126.....	RequestManager.py
128.....	שרת מחשוב
128.....	ComputeMachine.py
132.....	ComputeServerManager.py
134.....	Packets.py
139.....	ComputeClient.py



140	ArticleModel.py
144	ChatbotModel.py
148	Tokenizer.py
148	Transformer.py
148	PositionalEmbedding.py

## הכנת הנתונים, בניית ואימון המודל הראשון

חלק זה מורכב מהקובץ "transformer\_2500\_vocab.ipynb" וקובץ לאימון המודל באופן מקומי.

```
!pip install datasets

import math
import json
from datasets import load_dataset
import numpy as np
import tensorflow as tf
import tensorflow_datasets as tfds
from matplotlib import pyplot as plt
```

```
from google.colab import drive
drive.mount('/content/drive')
```

## קבועים

```
TOKENIZER_DIR =
'/content/drive/MyDrive/saved_models/tokenizer_2500_vocab.json'

CHECKPOINT_DIR =
'/content/drive/MyDrive/checkpoints/trained_checkpoint/checkpoint.ckpt'

DATASET_DIR =
'/content/drive/MyDrive/datasets/tokenized_articles'
LOG_DIR = '/content/drive/MyDrive/logs/training.log'

PAD_TOKEN = '<pad>'
START_TOKEN = '<s>'
END_TOKEN = '</s>'
# how many pairs we want to merge per epoch during tokenizer
training
MERGES_PER_EPOCH = 15

# model hyperparameters
BATCH_SIZE = 64
BUFFER_SIZE = 50_000
train_split = 0.8
validation_split = 0.1
test_split = 0.1
```

```

vocab_size = 2500
transformer_units = 6
d_model = 384
seq_length = 256
attention_heads = 8
dff = 1024 # transformer ffn hidden layer size
dropout_rate = 0.1
learning_rate = 1e-4

```

## הכנת הנתונים

### הורדת מאגר הנתונים

```

# TensorFlow has outdated checksums, and therefore we need to
# download the dataset from Hugging Face
dataset = load_dataset("cnn_dailymail", '3.0.0')

```

```

print(dataset)

output:
DatasetDict({
  train: Dataset({
    features: ['article', 'highlights', 'id'],
    num_rows: 287113
  })
  validation: Dataset({
    features: ['article', 'highlights', 'id'],
    num_rows: 13368
  })
  test: Dataset({
    features: ['article', 'highlights', 'id'],
    num_rows: 11490
  })
})

```

שרשור חלקי מאגר הנתונים. סיכומי הכתבות והמזהה שלהן לא נכללים מכיוון שלא נעשה מהם שימוש. לאחר מכן נעשה נרמול ראשוני בהפיכת כל האותיות המרכיבות של הכתבות לקטנות, מאחר והtokenizer לא תומך באותיות גדולות.

```

articles = list()

for split in ['train', 'validation', 'test']:
    articles.extend(dataset[split]['article'])
for i, e in enumerate(articles):
    articles[i] = e.lower()

```

## ייצירת ואימון הtokenizer

This is an implementation of the Byte-Pair-Encoding algorithm. In which we start with a base vocabulary of tokens. Then we find the most common pair of tokens and merge them into a new token. This process is repeated until we reach our desired vocabulary size. However, the encoding algorithm is slightly different from the normal BPE implementation. Instead of iterating over all possible merges and checking them against all tokens, the algorithm checks if adjacent tokens can be merged into a new token that exists in the vocabulary. This process is over 100X faster, in part due to the speed of set lookup. The resulting encodings are sometimes different, however from my observations they shouldn't lead to inferior results.

```

base_vocab = 'abcdefghijklmnopqrstuvwxyz! "$%&\' () *, -
./0123456789:;<?[\]\{\}\n '

class Tokenizer:
    """
    An implementation of a BPE (Byte-Pair-Encoding) tokenizer.

    The tokenizer allows us to create a token vocabulary size of
    our choosing while preventing Out-Of-Vocabulary problems. This
    is done by starting with a base vocabulary comprised of basic
    characters. The tokenizer then goes over the training corpus,
    finds the most frequent token pair, and merges them into a new
    token. This process is done until our desired vocabulary size
    is reached.

    Attributes:
        vocab_size (int): The maximum of the tokenizer's
        vocabulary.
        oov_token (str): The predefined Out-Of-Vocabulary token.
        tuple_delimiter (char): The delimiter used to separate
        tuple elements when

```

```

        serializing the tokenizer.
        vocab (str): the tokenizer's vocabulary.
        reserved_tokens (list[str]): Tokens reserved for other
purposes.
        merges (dict[tuple[str], str]): The token merges learned
during training.
    """

    def __init__(self, vocab_size: int, pad_token: str =
'<pad>', oov_token: str = '<oov>', reserved_tokens: list[str]
= list()) -> None:
    """
    Tokenizer constructor.

    Initializes the tokenizer's attributes and loads the base
vocabulary. Training
    the tokenizer is done by calling the ``train_on_corpus``
function, not by
    calling the constructor.

    Args:
        vocab_size: The maximum size of the tokenizer's
vocabulary.
        oov_token: The predefined Out-Of-Vocabulary token.
        reserved_tokens: Predefined tokens such as padding,
start and end.
    """
    self.vocab_size = vocab_size
    self.oov_token = oov_token
    self.tuple_delimiter = chr(0x1d)

    self.vocab = [pad_token] + [oov_token]
    for token in reserved_tokens:
        self.vocab.append(token)

    self.vocab += list(base_vocab)
    self.vocab_set = set(self.vocab)

    reserved_tokens.append(pad_token)
    reserved_tokens.append(oov_token)
    self.reserved_tokens = reserved_tokens

```

```

self.create_lookup_maps()

self.merges = dict()

def tokenize(self, input: str) -> list[int]:
    """
    Tokenizes a string using the tokenizer's learned merge
    table.

    The input is first split up into base tokens found in the
    ``base_vocab`` vocabulary. A loop then iterates over the
    tokens and checks whether the concatenation of adjacent pairs
    is in the vocabulary. If it is, the tokens are merged. Once
    the tokenizer can no longer merge tokens, the tokens are
    vectorized and returned.

    Args:
        input: The string we want to tokenize.

    Returns:
        A list containing the tokenized input in vector form.
    """

    split = [c for c in input if c in self.vocab_set]

    flag = True
    while flag:
        flag = False
        i = 0
        while len(split) - 1 > i:
            merge = split[i] + split[i + 1]
            if merge in self.vocab_set:
                split[i:i+2] = [merge]
                flag = True
            else:
                i += 1

        vectorized = list(map(lambda t: self.token_to_index[t],
                               split))

    return vectorized

```

```
def detokenize(self, input: list[int]) -> str:
    """
    Detokenizes a list of tokens in vector form into a string.

    Each token vector is mapped to its corresponding token.
    The tokens are then joined together to form the final string.

    Args:
        input: The list of token vectors we want to detokenize.

    Returns:
        A string derived from the input tokens.
    """
    # map each token vector, the index, to its string form
    devectorized = list(map(lambda i: self.index_to_token[i],
                             input))

    return ''.join(devectorized)

def train_on_corpus(self, corpus: list[str], verbose=False)
-> None:
    """
    Trains the tokenizer on a corpus.

    The corpus is first normalized, a process in which
    characters not present in the base vocabulary are removed. It
    is then pretokenized, meaning the corpus is split up into
    words and their frequencies to reduce computation later. After
    that, the words are split up into their base token
    representations. Finally we search for the most frequent token
    pair, merge them into a new token, add it to the vocabulary
    and merges dictionary, and update our word splits. This is
    done until our desired vocabulary size is reached.

    In order to reduce training time a number of pairs are
    merges in each "epoch". These pairs are picked by sorting the
    pair frequencies by highest lowest.

    Args:
```

```

        corpus: The corpus we want to train on. It's recommended
to preprocess the
        text before tokenizer training, but not necessary.
    """
    # normalization & pretokenization
    word_freq = dict()
    for e in corpus:
        for word in self.normalize_and_split(e):
            word_freq[word] = word_freq.get(word, 0) + 1

    # initial word split
    splits = { word:[c for c in word] for word in
word_freq.keys() }

    # merging
    while len(self.vocab) < self.vocab_size:
        pair_freq = self.calculate_pair_freq(word_freq, splits)
        best_merges = sorted(pair_freq.items(), key=lambda x: -
x[1])

        left = self.vocab_size - len(self.vocab)
        for i in range(min(MERGES_PER_EPOCH, left)):
            new_merge = best_merges[i][0]
            self.merges[new_merge] = new_merge[0] + new_merge[1]
            self.vocab.append(new_merge[0] + new_merge[1])
            self.vocab_set.add(new_merge[0] + new_merge[1])
            self.merge_pairs(new_merge[0], new_merge[1], splits)

        if verbose:
            percent = len(self.vocab) / self.vocab_size * 100
            print(f'\rCurrent vocab:
{len(self.vocab)}/{self.vocab_size} {percent:.2f}% ', end=' ')

        if verbose:
            print()

    self.create_lookup_maps()

    def normalize_and_split(self, text: str) -> list[str]:
        """
        Normalizes a text and splits it by whitespace.

```



Characters not found in the tokenizer's base vocabulary are filtered out. The text is then separated into individual words by whitespace while keeping the delimiter.

```

Args:
    text: The string we want to normalize and split.
"""
text = ''.join([c for c in text if c in base_vocab])
words = [' ' + word for word in text.split(' ') if word]
if len(words) > 0:
    words[0] = words[0][1:] # remove space from first
element

return words

def calculate_pair_freq(self, word_freq: dict[str, int],
splits: dict[str, list[str]]) -> dict[tuple[str, str], int]:
    """
    Calculates all pair frequencies in a pretokenized corpus.

    All token pairs of the pretokenized words are iterated
    over and their frequency tracked.

    Args:
        word_freq: The pretokenized words with their
        frequencies.
        splits: The tokens comprising the pretokenized words.

    Returns:
        A dictionary of all token pairs with their respective
        frequencies.
    """
    pair_freq = dict()
    for word, freq in word_freq.items():
        split = splits[word]
        for pair in zip(split, split[1:]): # zip adjacent
tokens
            pair_freq[pair] = pair_freq.get(pair, 0) + freq

    return pair_freq

```

```

def merge_pairs(self, t1: str, t2: str, splits: dict[str,
list[str]]) -> None:
    """
    Merges tokens comprising pretokenized words.

    All token pairs in the pretokenized words are compared
    against the merge tokens. If a match is found the tokens are
    compined and replaced with the merged token.

    Args:
        t1: The first token to merge.
        t2: The seconds token to merge.
        splits: The pretokenized words with their token splits.
    """
    for word, split in splits.items():
        i = 0
        while len(split) - 1 > i:
            if split[i] == t1 and split[i + 1] == t2:
                split[i:i+2] = [t1 + t2]
            else:
                i += 1

        splits[word] = split

def create_lookup_maps(self) -> None:
    """
    Creates mappings for token to index and index to token.

    The vocabulary is enumerated over and the appropriate key
    and values are placed in a map.
    """
    self.token_to_index = { token:i for i, token in
enumerate(self.vocab) }
    self.index_to_token = { i:token for i, token in
enumerate(self.vocab) }

def serialize_to_file(self, file_path: str) -> None:
    """
    Serializes the tokenizer and saves it to a file.

```

The tokenizer's attributes are turned into JSON form and then saved to a file. The merges map contains a tuple type which cannot be serialized to JSON. Therefore, we place the tuple elements in a string and separate them with the `self.tuple_delimiter` char.

```

Args:
    file_path: The file path for saving the serialized file.
"""
data = dict()
data['vocab_size'] = self.vocab_size
data['oov_token'] = self.oov_token
data['vocab'] = self.vocab
data['reserved_tokens'] = self.reserved_tokens
serialized_merges = { k[0]+self.tuple_delimiter+k[1] : v
for k, v in self.merges.items() }
data['merges'] = serialized_merges

with open(file_path, 'w') as f:
    json.dump(data, f)

def deserialize_from_file(self, file_path: str) -> None:
    """
    Deserializes the tokenizer from a serialized JSON file.

    The JSON file containing the serialized tokenizer is read
    from and the tokenizer's attributes are set. The tuples in the
    merges map are represented as a string with their elements
    separated by a delimiter. During deserialization we split this
    string by the delimiter and recreate the tuple.
    """
    with open(file_path, 'r') as f:
        data = json.load(f)
        self.vocab_size = data['vocab_size']
        self.oov_token = data['oov_token']
        self.vocab = data['vocab']
        self.vocab_set = set(self.vocab)
        self.reserved_tokens = data['reserved_tokens']
        serialized_merges = data['merges']
        self.merges = dict()

```

```

for k, v in serialized_merges.items():
    delimiter_idx = k.index(self.tuple_delimiter)
    key = (k[:delimiter_idx], k[delimiter_idx+1:])
    self.merges[key] = v

self.create_lookup_maps()

```

Train the tokenizer and serialize its state. Only the first 100,000 articles are using for training to reduce compute time while keeping good merging accuracy. Only two reserved tokens need to be defined, start of text and end of text, because the constructor has default definitions for padding and out of vocabulary tokens.

```

reserved_tokens = [START_TOKEN, END_TOKEN]
tokenizer = Tokenizer(vocab_size,
reserved_tokens=reserved_tokens)

tokenizer.train_on_corpus(articles[:100_000], verbose=True)

tokenizer.serialize_to_file(TOKENIZER_DIR)

```

Evaluating the tokenizer on an excerpt from an article not found in the training set. As we can see, commonly used words are encoded using a single token. Whereas rarer words are split up across multiple tokens. It can also be observed that the tokenizer can detokenize the tokens as intended.

```

tokenized = tokenizer.tokenize(' on one hand there are a lot
of people who do not understand what is going on')
print(tokenized)
for token in tokenized:
    print(f'{token}=>{tokenizer.detokenize([token])}')
print(tokenizer.detokenize(tokenized))

```

```

output:
[128, 332, 894, 367, 200, 69, 1087, 101, 352, 243, 301, 228,
542, 150, 23, 187, 389, 142, 776, 128]
128=> on
332=> one
894=> hand
367=> there
200=> are
69=> a

```

```

1087=> lot
101=> of
352=> people
243=> who
301=> do
228=> not
542=> und
150=>ers
23=>t
187=>and
389=> what
142=> is
776=> going
128=> on
on one hand there are a lot of people who do not understand
what is going on

```

Accessing the tokenizer's speed using Google Colab's built in timeit function. The sentence chosen for this task is an excerpt from an article not found in the training set with appropriate reserved tokens added.

```

corpus = ['<s>with that being said, there are many who think
that</s><pad>'] * 1000

%%timeit
for e in corpus:
    tokenizer.tokenize(e)

output:
47.2 ms ± 703 µs per loop (mean ± std. dev. of 7 runs, 10
loops each)

```

### יצירת מאגר הנתונים לאימון

The dataset pipeline is created by tokenizing the articles, splitting them up into sequence-length long chunk and packaging them up in tensorflow's Dataset object. Using a dataset object gives us more control and flexibility, primarily in the way the data is loaded into ram. In order to reduce the dataset's disk space, sequence-length plus one long chunks are stored. When loading the dataset, these chunks are turned into the X and Y values for the function.

```
tokenizer = Tokenizer(vocab_size)
tokenizer.deserialize_from_file(TOKENIZER_DIR)

for i, e in enumerate(articles):
    # overwrite articles with their tokenized form to reduce ram
    # usage
    articles[i] = tokenizer.tokenize(e)
    percent = (i + 1) / len(articles) * 100
    print(f'\rArticles tokenized: {i+1}/{len(articles)}
    {percent:.2f}%', end=' ')

dataset = tf.data.Dataset.from_tensor_slices(articles)
dataset.save(DATASET_DIR)
```

Split each article into 257 length chunks and append them to list. Articles are padded on their right side until they reach a multiple of 257. Start-of-text and end-of-text tokens are also added to the beginning and end of the article, before it is split up into chunks. The list of chunks is then converted to a tensorflow Dataset object and saved.

```
split_length = seq_length + 1
start_token_index = [2]
end_token_index = [3]

padded_data = list()

for e in articles:
    e = start_token_index + e + end_token_index
    e = np.array(e)
    splits = int(math.ceil(len(e) / split_length))
    padding_length = splits * split_length - len(e)

    e = np.pad(e, (0, padding_length))
    padded_data.extend(np.array_split(e, splits))

dataset = tf.data.Dataset.from_tensor_slices(padded_data)
dataset.save(DATASET_DIR)
```

### מנגנון התשומת לב

The scaled dot product attention is the core of the transformer architecture. The attention model is what allows the transformer to deal with large input sequences without exploding or vanishing gradients. It's comparable to a search engine where we

have a Query which gets compared to a list of Keys of Values. Values that have keys similar to the Query get a higher relevance score. The similarity between queries and keys is calculated using a scaled dot product:  $\text{softmax}(\frac{QK^T}{\sqrt{d_k}})$ . When one of the data dimensions in the queries vector and keys vector is large, it's dot product will also be large. When either or both are small the dot product is small. However, the total dot product grows as the data dimensions grow. Therefore, we need to scale the total dot product by a factor of  $\frac{1}{\sqrt{d_k}}$ .

```
def scaled_dot_product_attention(Q, K, V, mask):
    """
    Implementation of scaled dot product attention.

    The shape of the queries, values and keys is not known. We
    only care about the last two dimensions which must be of shape
    (tokens, d_k) for the queries and keys, and (tokens, d_v) for
    the values. This allows us to use the function both for normal
    attention and for multi head attention. Additionally, this
    means the function doesn't need batch dimension boilerplate.

    Args:
        Q: the queries. (... , seq_length, d_k)
        K: the keys. (... , seq_length, d_k)
        V: the values. (... , seq_length, d_v)
        mask: a boolean mask of elements not to attend to.
        (batches, heads, seq_length, seq_length)

    Returns:
        The scaled dot product attention of the queries, keys and
        values provided. If the input was comprised of multiple batches
        and heads the result will retain the number of batches and
        heads. (... , seq_length, d_v)
    """
    # matmul transposition uses `tf.linalg.matrix_transpose`
    # which only transposes
    # the inner-most 2d matrix
    x = tf.matmul(Q, K, transpose_b=True) # (... , seq_length,
    seq_length)
```

```

    # scale the dot product with of a factor of 1 over the
    standard deviation,
    # therefore the standard deviation is now unit
    d_k = Q.shape[-1]
    x /= d_k**0.5

    # apply look-ahead and padding masks, ones represent values
    we want to keep
    replace_with = tf.expand_dims(tf.ones(x.shape[-1:],
dtype=tf.float32) * -1e9, axis=0)
    x = tf.where(mask, x, replace_with)

    # apply the softmax functin across each token, horizontally
    x = tf.nn.softmax(x, axis=-1)

    x = tf.matmul(x, V) # (... , seq_length, d_v)

    return x

```

This multi-head attention architecture is identical to the one proposed in the "Attention is all you need" paper. The inputs are then split up into groups of size  $\frac{d_{model}}{heads}$ . Each group is projected using learned projection weights into keys, queries and values, which are then fed into a scaled dot product attention. The heads' results are then concatenated across the last dimension and multiplied by a learned projection weight  $O_w$ . Dropout is applied to prevent overfitting.

```

class Multi_Head_Attention(tf.keras.layers.Layer):
    """
    Implementation of multi head attention. This model is
    identical to the one proposed in the "Attention is all you
    need" paper.

    Attributes:
        supports_masking: tells tensorflow that this layer
        supports masking.
        d_model: the model's embedding dimension size.
        heads: the number of attention heads to be used.
        d_k: the size of keys and queries vectors.
        d_v: the size of the value vector.
        Q_w: the projection metrix for calculating queries,
        consists of learned parameters.

```



```

    K_w: the projection metrix for calculating keys, consists
of learned parameters.
    V_w: the projection metrix for calculating values,
consists of learned parameters.
    Q_w: the projection metrix for concatenating the attention
heads, consists of learned parameters.
    dropout: the dropout layer.
"""

def __init__(self, heads, d_model, dropout_rate):
    """
    Creates the Multi_Head_Attention object.

    Args:
        heads: the number of attention heads we want to use.
        d_model: the model's embedding dimension.
        dropout_rate: the attention mechanism's dropout rate.
    """
    super().__init__()
    self.supports_masking = True

    if d_model % heads != 0:
        raise ValueError('Model embedding dimensions must be
divisible by the number of attention heads!')

    self.d_model = d_model
    self.heads = heads

    # in this case d_k and d_v are equal, however that's just
for simplicity's sake
    self.d_k = d_model // heads
    self.d_v = d_model // heads

    # Xavier weight initialization
    seed = np.random.randint(1e9, size=1)[0]
    initializer = tf.keras.initializers.GlorotNormal(seed)
    # we need to add a 1 for the batch dimension, tensorflow's
matmul doesn't do this for us
    self.Q_w = tf.Variable(initializer(shape=(1, heads,
d_model, self.d_k)))
    self.K_w = tf.Variable(initializer(shape=(1, heads,
d_model, self.d_k)))

```

```

        self.V_w = tf.Variable(initializer(shape=(1, heads,
d_model, self.d_v)))
        self.O_w = tf.Variable(initializer(shape=(1, heads *
self.d_v, d_model)))

        self.dropout = tf.keras.layers.Dropout(dropout_rate)

    def call(self, inputs, mask=None):
        """
        Applies the multi-head self attention to a given input.
        Called during forward propagation and inference.

        Args:
            inputs: the layer's inputs, must be in the shape
            (batches, tokens, d_model).
            mask: an additional mask to ignore padding tokens, if no
            mask is provided only a causal mask will be applied.
        """

        # add heads dimension to inputs
        inputs = tf.expand_dims(inputs, axis=1) # (batches, 1,
tokens, d_model)

        Q = tf.matmul(inputs, self.Q_w) # (batches, heads, tokens,
d_k)
        K = tf.matmul(inputs, self.K_w) # (batches, heads, tokens,
d_k)
        V = tf.matmul(inputs, self.V_w) # (batches, heads, tokens,
d_v)

        # create masks
        seq_len = inputs.shape[-2]
        full_mask = tf.cast(np.tri(seq_len), dtype=tf.bool)
        if mask is not None:
            mask = tf.expand_dims(mask, axis=1) # (batches, 1,
seq_len)
            mask = tf.expand_dims(mask, axis=1) # (batches, 1, 1,
seq_len)
            mask = tf.tile(mask, tf.constant([1, 1, seq_len, 1])) #
(batches, heads, seq_len, seq_len)

```

```

    full_mask = full_mask & mask # (batches, heads, seq_len,
seq_len)

    x = scaled_dot_product_attention(Q, K, V, full_mask) #
(batches, heads, tokens, d_v)

    # concatenate heads
    batches, heads, tokens, d_v = x.shape
    # this line is the root of a bug which took me a week to
find
    x = tf.transpose(x, perm=[0, 2, 1, 3])
    # batches could be None, we therefore pass -1 so that the
batch size is calculated at runtime
    x = tf.reshape(x, (-1, tokens, heads*d_v)) # (batches,
tokens, d_model)

    x = self.dropout(x)

    x = tf.matmul(x, self.O_w) # (batches, tokens, d_model)

    return x

```

### שכבת החלחול קדימה

Each transformer block contains one fully connected feed forward network. The network is comprised of one dense layer with a relu non-linearity, followed by a dense layer of size  $d_{model}$ . Dropout is applied to prevent overfitting.

```

class FeedForwardNetwork(tf.keras.layers.Layer):
    """
    A simple implementation of the feedforward layer found in
    the "Attention is all you need" paper. The paper suggests
    implementing the layer using a convolution layer with a kernel
    of one, however this implementation is based on element-wise
    dense layers.

    Attributes:
        supports_masking: tells tensorflow that this layer
        supports masks.
        dense1: the first dense layer in the FFN.
        dense2: the second dense layer in the FFN
    """

```

```

        dropout: the dropout layer used to prevent overfitting.
    """

    def __init__(self, dff, d_model, dropout_rate):
        """
        Creates the FeedForwardNetwork object.

        Args:
            dff: the size of the hidden dense layer.
            d_model: the embedding dimension of the model.
            dropout_rate: the dropout rate used.
        """
        super().__init__()
        self.supports_masking = True

        self.dense1 = tf.keras.layers.Dense(dff,
activation='relu')
        self.dense2 = tf.keras.layers.Dense(d_model)
        self.dropout = tf.keras.layers.Dropout(dropout_rate)

    def call(self, inputs):
        """
        Applies the feed forward network. Called during forward
        propagation and inference.

        Args:
            inputs: the layer's inputs, must of of shape (batches,
seq_length, d_model)
        """
        x = self.dense1(inputs)
        x = self.dense2(x)
        x = self.dropout(x)

        return x

```

#### יחידת הטרנספורמר

The transformer block encapsulates the attention mechanisms and feed forward network along with layer normalization and residual connections into one block. This

block's goal is to extract and refine information from the input so that it can be used in the other layers of the network. Encasing the logic into one block allows for easy model configuration. The blocks can be layered atop each other with ease.

```
class TransformerBlock(tf.keras.layers.Layer):
    """
    Implementation of the full transformer block, consisting of
    attention mechanisms and a feed forward network along with
    layer normalization and residual connections.

    Attributes:
        supports_masking: tells tensorflow that this layer supports
        masks.
        layer_norm_1: the first normalization layer.
        residual_connection: the residual connection component. The
        same object is used twice as it doesn't have learnable
        parameters.
        layer_norm_2: the second normalization layer.
        feed_forward_network: the transformer's feed forward
        network.
    """

    def __init__(self, attention_heads, dff, d_model,
                 dropout_rate):
        """
        Creates a transformer object.

        Args:
            attention_heads: the desired number of heads in the
            attention mechanism.
            dff: the size of the hidden layer in the feed forward
            network.
            dropout_rate: the dropout rate for the various parts of
            the transformer.
        """
        super().__init__()
        self.supports_masking = True

        self.layer_norm_1 = tf.keras.layers.LayerNormalization()
        self.multi_head_attention =
        Multi_Head_Attention(attention_heads, d_model, dropout_rate)
        self.residual_connection = tf.keras.layers.Add()
```

```

self.layer_norm_2 = tf.keras.layers.LayerNormalization()
self.feed_forward_network = FeedForwardNetwork(dff,
d_model, dropout_rate)

def call(self, inputs):
    """
    Applies the transformer logic to an input. Used during
    forward propagation and inference.

    Args:
        inputs: the input matrix, must be of shape (batches,
seq_length, d_model)
    """
    norm_1_out = self.layer_norm_1(inputs)
    attention_out = self.multi_head_attention(norm_1_out)
    residual_1_out = self.residual_connection([norm_1_out,
attention_out])

    norm_2_out = self.layer_norm_2(residual_1_out)
    ffn_out = self.feed_forward_network(norm_2_out)
    residual_2_out = self.residual_connection([norm_2_out,
ffn_out])

    return residual_2_out

```

## קידוד מיקום

Positional embedding as proposed in the "Attention is all you need" paper.

```

def positional_encoding(length, depth):
    """
    Creates the positional encoding matrix from the sinusoidal
    encoding formulas.

    Args:
        length: number of possible sequence positions.
        depth: the depth of the encoding, equal to the token
embedding dimension.
    """

```

```

depth = depth/2

positions = np.arange(length)[: , np.newaxis] # (seq, 1)
depths = np.arange(depth)[np.newaxis, :] / depth # (1,
depth)

angle_rates = 1 / (10000**depths) # (1, depth)
angle_rads = positions * angle_rates # (pos, depth)

pos_encoding = np.concatenate(
    [np.sin(angle_rads), np.cos(angle_rads)],
    axis=-1)

pos_encoding = np.expand_dims(pos_encoding, axis=0)

return tf.cast(pos_encoding, dtype=tf.float32)

```

```

class PositionalEmbedding(tf.keras.layers.Layer):
    """
    Implementation of a positional embedding layer based on the
    sinusoidal encoding formulas from the "Attention is all you
    need" paper.

    Attributes:
        supports_masking: tells tensorflow that this layer
        supports masking.
        pe_table: the table containing the positional encoding
        values.
    """

    def __init__(self, max_length, d_model):
        """
        Creates a PostionalEmbedding object.

        Args:
            max_length: the maximum sequence length.
            d_model: the token embedding dimension.
        """
        super().__init__()
        self.supports_masking = True

```

```

self.pe_table = positional_encoding(max_length, d_model)

def call(self, inputs):
    """
    Adds the positional embedding values to an input. Called
    during forward propagatin and inference.

    Args:
        input: the input matrix, must be of shape (batches,
        seq_length, d_model)
    """
    batches, seq_length, d_model = inputs.shape

    # scale up token embeddings so that they have more
    influence than positional embedding
    x = inputs * np.sqrt(d_model, dtype=np.float32)

    x = x + 0.1 * self.pe_table[:, :seq_length, :]

    return x

```

### קוד אימון המודל

קוד לא נכתב בסביבת Google Colab ושימש רק להרצת תהליך האימון. מחלקת callback מיוחדת נכתבה כדי לשמור את ערכי ההפסד עבור כל צעד אימון, במחשבה שהמודל יעבור מספר epochs קטן בכדי להשתמש בגרף מבוסס epochs. אם זאת, נמצא כי זמן האימון יחסית קטן ומאפשר למספר epochs מספק. המחלקות לשכבות השונות שנכתבו מופיעות בקבצים שונים לשם סדר. תוכן הקבצים זהה ליישום שהוצג לעיל, ולכן אין טעם לצרף את הקבצים.

```

import os.path
import numpy as np
import tensorflow as tf
from transformer import TransformerBlock,
Multi_Head_Attention, FeedForwardNetwork
from positionalembdding import PositionalEmbedding

```



```

CHECKPOINT_DIR = 'E:/language
model/article_big/checkpoint.ckpt'
DATASET_DIR = 'E:/language model/dataset/tokenized_articles'
LOG_DIR = 'E:/language model/log/training.log'

PAD_TOKEN = '<pad>'
START_TOKEN = '<s>'
END_TOKEN = '</s>'
MERGES_PER_EPOCH = 15 # how many pair we want to merge per
epoch during tokenizer training

# model hyperparamaters
BATCH_SIZE = 48
BUFFER_SIZE = 50_000
train_split = 0.8
validation_split = 0.1
test_split = 0.1
vocab_size = 2500
transformer_units = 6
d_model = 384
seq_length = 256
attention_heads = 8
dff = 1024 # transformer ffn hidden layer size
dropout_rate = 0.1
learning_rate = 1e-4

class LossLoggerCallback(tf.keras.callbacks.Callback):

    def __init__(self, file_path, step=1):
        super().__init__()
        self.file_path = file_path
        self.step = step
        self.buffer = list()

    def on_train_batch_end(self, batch, logs=None):
        loss = logs.get('loss')
        accuracy = logs.get('accuracy')
        if loss and accuracy:
            self.buffer.append((self.step, loss, accuracy))

```

```

        self.step += 1

    def on_epoch_end(self, epoch, logs=None):
        output = [f'\n{step}, {loss}, {accuracy}' for step, loss,
accuracy in self.buffer]
        with open(self.file_path, 'a') as f:
            f.writelines(output)

        self.buffer.clear()

def load_dataset():
    dataset = tf.data.Dataset.load(DATASET_DIR)

    preprocess = lambda x: (x[:, :-1], x[:, 1:])
    dataset = (dataset
                .batch(BATCH_SIZE)
                .map(preprocess, tf.data.AUTOTUNE)
                .prefetch(buffer_size=tf.data.AUTOTUNE))

    size = len(dataset)
    training_dataset = dataset.take(int(size * train_split))
    dataset = dataset.skip(int(size * train_split))
    validation_dataset = dataset.take(int(size *
validation_split))
    testing_dataset = dataset.skip(int(size * validation_split))

    print('training dataset size:', len(training_dataset))
    print('validation dataset size:', len(validation_dataset))
    print('testing dataset size:', len(testing_dataset))

    return training_dataset, validation_dataset, testing_dataset

def main():
    inputs = tf.keras.Input(shape=(seq_length,))
    # token embedding layer
    x = tf.keras.layers.Embedding(vocab_size, d_model,
mask_zero=True)(inputs)
    # positional embedding layer
    x = PositionalEmbedding(5000, d_model)(x)

```

```

for _ in range(transformer_units):
    x = TransformerBlock(attention_heads, dff, d_model,
dropout_rate)(x)

    x = tf.keras.layers.LayerNormalization()(x)

    outputs = tf.keras.layers.Dense(vocab_size)(x)

    model = tf.keras.Model(inputs=inputs, outputs=outputs)

    model.compile(loss=tf.keras.losses.SparseCategoricalCrossent
ropy(from_logits=True),
optimizer=tf.keras.optimizers.Adam(learning_rate=lear
ning_rate),
metrics=['accuracy'])

    model.load_weights(CHECKPOINT_DIR)

    print(model.summary())

    training_dataset, validation_dataset, testing_dataset =
load_dataset()

    log_callback =
tf.keras.callbacks.CSVLogger('C:/Users/Arzi0/OneDrive/Desktop/
AI final project/training/article/log.csv', append=True)
    loss_step_log_callback =
LossLoggerCallback('C:/Users/Arzi0/OneDrive/Desktop/AI final
project/training/article/log_loss.txt')
    checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
        filepath=CHECKPOINT_DIR,
        monitor='val_accuracy',
        mode='max',
        save_best_only=True,
        save_weights_only=True)

    callbacks = [log_callback, checkpoint_callback,
loss_step_log_callback]

    model.fit(training_dataset,
validation_data=validation_dataset,
epochs=10,

```

```
callbacks=callbacks)

if __name__ == '__main__':
    main()
```

## הכנת הנתונים, בניית ואימון המודל שני

חלק זה מורכב מקובץ "GPT\_text\_generation.ipynb" שמכיל את הקוד לטעינה ואימון המודל, וקטעי קוד ליצירת והכנת מאגר הנתונים.

### יצירת והכנת מאגר הנתונים

כזכור מאגר הנתונים הזה מורכב משושה תתי מאגרים. המאגר הראשון הוא מאגר קיים שנבנה כחלק בפרויקט alpaca של אוניברסיטת סטנפורד. בנוסף לכך, נבנו שני מאגרים עם שאלות המתאימות לנושא הקיימות. שאלות אלה נאספו באמצעות מודל שיחתי קיים ונשמרו בקבצי json.

```
import json

prompt_format = 'Below is an instruction that describes a task. Write a response that appropriately completes the request.\n\n### Instruction:\n{question}\n\n### Response:\n{answer}'
full_dataset = list()

dataset1 = json.load(open('alpaca_data.json', 'r'))
for e in dataset1:
    question = e['instruction']
    if e['input'] != '':
        instruction += '\n\n' + e['input']

    full_dataset.append(prompt_format.format(question=question,
        answer=e['output']))

dataset2 = json.load(open('sustainability_question.json'))
for e in dataset2:
    full_dataset.append(prompt_format.format(question=e['question'],
        answer=e['answer']))
```

```
dataset3 = json.load(open('personality_questions.json'))
for e in dataset3:
    full_dataset.append(prompt_format.format(question=e['question'], answer=e['answer']))

json.dump(full_dataset, open('dataset.json', 'w'))
```

```
import keras_nlp
import json

tokenizer =
keras_nlp.models.OPTTokenizer.from_preset('opt_125m_en')

untokenized_dataset =
json.load(open('untokenized_dataset.json', 'r'))

dataset = {
    'x': list(),
    'y': list(),
    'padding_mask': list()
}

for raw in untokenized_dataset:
    e = tokenizer(raw).numpy().tolist()

    if len(e) > 255:
        continue

    e = [0] + e + [2]
    padding_mask = [1] * min(256, len(e))
    padding_mask += [0] * (256 - len(padding_mask))
    e += [1] * (257 - len(e))
    x = e[:-1]
    y = e[1:]
    dataset['x'].append(x)
    dataset['y'].append(y)
    dataset['padding_mask'].append(padding_mask)

json.dump(dataset, open('dataset.json', 'w'))
```

## טעינת המודל ואימונו

נעשה שימוש בספריית עזר בשם keras\_nlp שאינה מופיע בקי, לכן כדי להוריד אותה נדרש לצרף קישור לספרייה שנמצא ב-GitHub. במהלך כתיבת הפרויקט ספרית tensorflow התעדכנה, דבר שהוביל לאי התאמה בין הספריות. לכן נאלצתי להשתמש בגרסה ישנה של tensorflow. הדבר לא מוביל להתנגשות עם חלקים אחרים של הפרויקט מאחר וביישום כל מודל רץ בסביבה שונה ואף במחשב שונה.

```
!pip install tensorflow=={2.15}
!pip install git+https://github.com/keras-team/keras-
nlp.git@v0.9.3 -q
```

```
import os

os.environ["KERAS_BACKEND"] = "tensorflow"

import keras_nlp
import keras
import tensorflow as tf
import numpy as np
import time
import json

keras.mixed_precision.set_global_policy("mixed_float16")

# CHECKPOINT_DIR =
# '/content/drive/MyDrive/final_model/checkpoint/checkpoint.ckpt
# '
CHECKPOINT_DIR =
'/content/drive/MyDrive/final_model/weights/checkpoint.weights
.h5'
DATASET_DIR =
'/content/drive/MyDrive/final_model/dataset/dataset.json'
LOG_DIR =
'/content/drive/MyDrive/final_model/logs/training.csv'
```

```

LOSS_LOG_DIR =
'/content/drive/MyDrive/final_model/logs/loss.txt'

TRAINING_SPLIT = 0.8
VALIDATION_SPLIT = 0.1
TESTING_SPLIT = 0.1

EPOCHS = 10
BATCH_SIZE = 5

```

```

from google.colab import drive
drive.mount('/content/drive')

```

הורדת המודל הקיים והtokenizer שעליו התאמן. הפרמטר `preprocessor` שהוגדר ל-`None` גורם לכך שהמודל יהיה ללא שכבה שעושה את תהליך הtokenization מראש. השכבה אמנם נוחה, ניתן פשוט להעביר מחרוזת ולקבל קלט, אולם היא מעיטה משמעותית את תהליך הלמידה. לכן הוחלט להסירה.

```

model =
keras_nlp.models.OPTCausalLM.from_preset("opt_125m_en",
preprocessor=None)

model.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(
    from_logits=True),
              optimizer=tf.keras.optimizers.Adam(learning_rate=
1e-4),
              metrics=['accuracy'])

tokenizer =
keras_nlp.models.OPTTokenizer.from_preset("opt_125m_en")

```

```

dataset = json.load(open(DATASET_DIR, 'r'))
dataset_length = len(dataset['x'])
split = int(dataset_length * (TRAINING_SPLIT +
VALIDATION_SPLIT))

x = {
    "token_ids": np.array(dataset['x'][:split]),

```

```

        "padding_mask": np.array(dataset['padding_mask'][:split]),
    }

    y = np.array(dataset['y'][:split])
    sw = np.array(dataset['sw'][:split])

```

```

checkpoint_callback =
tf.keras.callbacks.ModelCheckpoint(filepath=CHECKPOINT_DIR,
save_weights_only=True)
log_callback =
LossLoggerCallback('/content/drive/MyDrive/final_model/logs/log.txt')
callbacks = [checkpoint_callback, log_callback]

history = model.fit(x=x,
                    y=y,
                    validation_split=VALIDATION_SPLIT,
                    sample_weight=sw,
                    epochs=1,
                    batch_size=20,
                    callbacks=callbacks)

```

יישום

main.py

```

from RequestsManager import RequestManager
from ComputeServerManager import ComputeServerManager
from threading import Thread
import WebServer

def main():
    """
    Initializes the backend process. This process contains the
    web server, request manager and compute server.
    :return:
    """

```



```

    compute_server_manager = ComputeServerManager('0.0.0.0',
1337)
    compute_server_manager.start()

    request_dispatcher = RequestManager(compute_server_manager)
    dispatcher_thread =
Thread(target=request_dispatcher.dispatch_requests)
    dispatcher_thread.start()

    WebServer.run(request_dispatcher, compute_server_manager)

if __name__ == '__main__':
    main()

```

## שרת אינטרנט

## WebServer.py

```

from flask import Flask, render_template, request, redirect,
flash, copy_current_request_context
from flask_login import LoginManager, UserMixin, login_user,
login_required
from flask_sqlalchemy import SQLAlchemy
from flask_socketio import SocketIO, emit
from Requests import QuestionRequest, ArticleRequest

# ===== Website backend configuration =====

app = Flask(__name__)
app.config['SECRET_KEY'] =
'37806fa4a1c653c133874fc9e3ef576ddc83fbe294fc9f8ab97010421784e
acf'

# database configuration
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///db.sqlite'
db = SQLAlchemy()

class Users(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)

```

```

    username = db.Column(db.String(100), unique=True,
nullable=False)
    password = db.Column(db.String(100), nullable=False)
    isAdmin = db.Column(db.Boolean, default=False,
nullable=False)

db.init_app(app)

with app.app_context():
    db.create_all()

# websocket support
socketio = SocketIO(app)

# login manager configuration
login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'

@login_manager.user_loader
def loader_user(user_id):
    return Users.query.get(user_id)

# global objects
request_dispatcher = None
compute_server_manager = None

# ===== Route definitions =====

@app.route('/login', methods=['GET', 'POST'])
def login():
    """
    Handles login attempts.

    When someone attempts to log into the website via /login,
    the requests will be routed to this function. The form's
    information is available via the request variable.
    """

```

```
if request.method == 'POST':
    form_username = request.form['username']
    form_password = request.form['password']

    user =
Users.query.filter_by(username=form_username).first()

    if user is not None and user.password == form_password:
        login_user(user)
        flash('logged in!')
        return redirect('index')
    else:
        flash('incorrect credentials')

    return render_template('login.html')

@app.route('/')
@app.route('/index')
@login_required
def index():
    return render_template('index.html')

@app.route('/chatbot')
@login_required
def chatbot():
    return render_template('chatbot.html')

@app.route('/article-generator')
def article_generator():
    return render_template('article_generator.html')

@app.route('/compute-server')
@login_required
def compute_server():
    # a list of current compute machines is passed to the
    template
    return render_template('compute_server.html',
server_list=compute_server_manager.compute_machines)
```

```
# ===== Web socket traffic =====

@socketio.on('generate_chatbot_response')
@login_required
def generate_chatbot_response(data):
    """
    Handles websocket requests to generate a chatbot response.

    Tokens are returned to the client one by one until an end-
    of-text token is received.

    :param data: contains the request information.
    """
    sid = request.sid

    r = QuestionRequest(data['prompt'], float(data['temp']),
float(data['top_p']))

    @copy_current_request_context
    def handle_token(token):
        emit('response', {'token': token})
        if token == '</s>':
            request_dispatcher.remove_request(sid)

    r.set_token_handler(handle_token)

    request_dispatcher.add_request(sid, r)

@socketio.on('generate_article')
@login_required
def generate_article(data):
    """
    Handles websocket requests to generate articles.

    Tokens are returned to the client one by one until an end-
    of-text token is received.

    :param data: contains the request information.
    """
    sid = request.sid
```

```

    r = ArticleRequest(data['prompt'], float(data['temp']),
float(data['top_p']))

    @copy_current_request_context
    def handle_token(token):
        emit('response', {'token': token})
        if token == '</s>':
            request_dispatcher.remove_request(sid)

    r.set_token_handler(handle_token)

    request_dispatcher.add_request(sid, r)

@socketio.on('stop_generating')
@login_required
def stop_generating(data):
    """
    Handles websocket requests to stop generating a result.
    """
    request_dispatcher.remove_request(request.sid)

def run(request_dispatcher_param,
compute_server_manager_param):
    global request_dispatcher, compute_server_manager
    request_dispatcher = request_dispatcher_param
    compute_server_manager = compute_server_manager_param
    app.run(host='0.0.0.0')

```

login.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
</head>
<body>
    <form method="post">

```

```

{% with messages = get_flashed_messages() %}
{% if messages %}
    <ul class=flashes>
        {% for message in messages %}
            <li>{{ message }}</li>
        {% endfor %}
    </ul>
{% endif %}
{% endwith %}
<label for="username">Username:</label><br>
<input type="text" id="username" name="username"><br>

<label for="password">password:</label><br>
<input type="password" id="password" name="password"><br>

<br>

<input type="submit" value="Submit">
</form>
</body>
</html>

```

base.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>AI project</title>
    <link rel="stylesheet" href="{{ url_for('static',
filename='base.css') }}">
    <link rel="stylesheet" href="{% block stylesheet %}{%
endblock %}">
</head>
<body>
    <div class="navigation-header">
        <h1>{% block header %}{% endblock %}</h1>
        <div class="navigation-button-container">
            <a href="{{ url_for('chatbot') }}"><button
class="navigation-button">Chatbot</button></a>

```

```

    <a href="{{ url_for('article_generator') }}"><button
class="navigation-button">Article generator</button></a>
    <a href="{{ url_for('compute_server') }}"><button
class="navigation-button">Compute server
configuration</button></a>
    <a href="{{ url_for('index') }}"><button
class="navigation-button">About</button></a>
    </div>
</div>

{% block content %}{% endblock %}
</body>
</html>

```

base.css

```

body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f0f0f0;
}

.navigation-header {
    background-color: #007bff;
    color: #fff;
    padding: 20px;
    text-align: center;
}

.navigation-button-container {
    display: inline-block;
}

.navigation-button {
    padding: 8px 20px;
    background-color: #fff;
    color: #007bff;
    border: 1px solid #007bff;
    border-radius: 5px;
    cursor: pointer;
}

```

```

    margin-right: 10px;
}

.navigation-button:hover {
    background-color: #0056b3;
    color: #fff;
}

```

index.html

```

{% extends 'base.html' %}

{% block stylesheet %}{{ url_for('static',
filename='index.css') }}{% endblock %}

{% block header %}About the project{% endblock %}

{% block content %}
    <div class="main-content">
        <div class="chatbot-description">
            <h2>About this project</h2>
            <p>
                The purpose of this website is to present two Natural
                Language Processing (NLP) neural networks that are part of a
                final project in an artificial intelligence course. Both
                models are powered by the transformer architecture found in
                other Generative Pretrained Transformer (GPT) models. In order
                to run these models in parallel and allow multiple users to
                run them at the same time, a network of computational servers
                was implemented. Each server hosts a model and is open to
                inference requests, which are dispatched by a central server.
            <br>
            <br>
            The first model was trained to write news articles
            based on the input a user gives it. The input can be the first
            word of the article, first line, first paragraph or be left
            blank. Dataset wise, the model was trained on some 312,000
            news articles sourced from CNN. It boasts 6 transformer
            decoder units stacked atop each other, as well as embedding,
            encoding and dense layers. The model has a total of 10 million

```



trainable parameters, and was trained on an RTX 2060 for around 40 hours.

<br>

<br>

The second model is an upscaled version of the first model, containing 12 transformer blocks and higher dimensionality for a grand total of 125 million trainable parameters. Because of its size, training such a model from scratch is infeasible. Therefore, a pretrained model was obtained from the Open Pretrained Transformers project, which was trained on a general dataset. In order to fine tune the model for question answering a dataset was created by combining Stanford's LLaMa dataset with custom generated questions, for a total of 54,000 question-answer pairs. The model was trained for 8 hours on a T4 GPU provided by Google Colab.

</p>

</div>

</div>

{% endblock %}

index.css

```
.main-content {
  max-width: 800px;
  margin: 20px auto;
  padding: 20px;
  background-color: #fff;
  border-radius: 8px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

.chatbot-description {
  margin-top: 20px;
}

.chatbot-description h2 {
  color: #007bff;
}

.chatbot-description p {
```

```

    line-height: 1.5;
}

```

chatbot.html

```

{% extends 'base.html' %}

{% block stylesheet %}{{ url_for('static',
filename='chatbot.css') }}{% endblock %}

{% block header %}Chatbot interface{% endblock %}

{% block content %}
    <div class="chat-container">
        <div class="chat-box" id="chat-box"></div>
        <input type="text" id="user-input" class="prompt-input"
placeholder="Type your message...">

        <div class="button-container">
            <button onclick="start_generating()" id="send-
button">Send</button>
            <button onclick="stop_generating()" id="stop-button"
style="background: grey" disabled>Stop</button>
        </div>
        <label for="temp">Temperature:</label>
        <input type="number" id="temp" value="1" min="0.1" max="5"
step="0.1">
        <label for="top_p">Top_p:</label>
        <input type="number" id="top_p" value="1" min="0.1"
max="5" step="0.1">
        </div>

        <script
src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.3.2/so
cket.io.js"></script>
        <script src="{{ url_for('static', filename='chatbot.js')
}}"></script>
    {% endblock %}

```

## chatbot.css

```
.chat-container {
  max-width: 1000px;
  margin: 50px auto;
  padding: 20px;
  background-color: #fff;
  border-radius: 8px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

.chat-box {
  min-height: 500px;
  max-height: 600px;
  overflow-y: auto;
  padding: 10px;
  border: 1px solid #ddd;
  border-radius: 5px;
  margin-bottom: 10px;
}

.chat-box .message {
  margin-bottom: 10px;
}

.chat-box .message.user .message-content {
  background-color: #dddddd;
  padding: 8px 12px;
  border-radius: 5px;
  display: inline-block;
  word-wrap: break-word;
}

.chat-box .message.chatbot .message-content {
  background-color: #4caf50;
  color: white;
  padding: 8px 12px;
  border-radius: 5px;
  display: inline-block;
  word-wrap: break-word;
}

.prompt-input {
```

```

width: calc(100% - 200px);
padding: 8px;
border: 1px solid #ddd;
border-radius: 5px;
margin-right: 10px;
}

.button-container {
  display: inline-block;
}

button {
  padding: 8px 20px;
  background-color: #007bff;
  color: #fff;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  margin-right: 10px;
}

button:hover {
  background-color: #0056b3;
}

```

chatbot.js

```

// create a socket with the server in order to communicate
var socket = io.connect('ws://' + document.domain + ':' +
location.port);

var chatBox = document.getElementById('chat-box');
var userInput = document.getElementById('user-input');
var sendButton = document.getElementById('send-button');
var stopButton = document.getElementById('stop-button');
var tempInput = document.getElementById('temp')
var topPInput = document.getElementById('top_p')

isGenerating = false;
currentResponseContainer = null;

```

```
function start_generating() {  
  /**  
    Called when the user click the send button. Checks that the  
    input is valid and sends the prompt to the server. Creates a  
    new message in the chatbox for the user's question and the  
    chatbot's response.  
  */  
  var input = userInput.value;  
  var temperature = tempInput.value;  
  var top_p_val = topPInput.value;  
  
  if (input.trim() == '')  
    return;  
  
  userInput.value = '';  
  
  onGenerationStart();  
  
  // create new messages, one for the user's question and the  
  // other for the chatbot's response  
  addMessage('user', input);  
  currentResponseContainer = addMessage('chatbot', '')  
  
  // send the prompt and inference config to the server  
  socket.emit('generate_chatbot_response', {prompt: input,  
    temp: temperature, top_p: top_p_val});  
};  
  
function stop_generating() {  
  /**  
    Called when the user clicks the stop button. Tells the  
    server to stop generating a response and allows the user to  
    send a new question.  
  */  
  if (!isGenerating)  
    return;  
  
  onGenerationEnd();  
  socket.emit('stop_generating', {user_id: 'dang'});  
}  
  
socket.on('response', function(data) {
```

```

/**
 * Handles responses from the server. Every token generated and
 * sent by the server will call this event.
 */
if (data['token'] == '</s>' || currentResponseContainer ==
null) {
    onGenerationEnd();
    return;
}

// add the new token to the current chatbot message
currentResponseContainer.innerHTML += data['token'];
});

function addMessage(sender, message) {
    /**
     * Renders a new message in the chatbox. This function is used
     * both for user messages and chatbot messages.
     */
    var messageElement = document.createElement('div');

    // add the message element's classes, the sender class is
    // used to ascertain whether its a user message or a chatbot
    // message
    messageElement.classList.add('message', sender);

    var senderSpan = document.createElement('span');
    senderSpan.className = 'message-sender';
    senderSpan.innerHTML = sender;
    messageElement.appendChild(senderSpan);

    messageElement.appendChild(document.createElement('br'));

    var contentSpan = document.createElement('span');
    contentSpan.className = 'message-content';
    contentSpan.innerHTML = message;
    messageElement.appendChild(contentSpan);

    chatBox.appendChild(messageElement);
    chatBox.scrollTop = chatBox.scrollHeight;
}

```

```

    return contentSpan;
}

function onGenerationStart() {
    /**
     * Updates the send/stop buttons when the chatbot starts
     * generating a response.
     */
    isGenerating = true;

    stopButton.disabled = false;
    stopButton.style.background = '#007bff';
    sendButton.disabled = true;
}

function onGenerationEnd() {
    /**
     * Updates the send/stop buttons when the chatbot stops
     * generating a response.
     */
    isGenerating = false;

    stopButton.disabled = true;
    stopButton.style.background = 'grey';
    sendButton.disabled = false;
}

```

article\_generator.html

```

{% extends 'base.html' %}

{% block stylesheet %}{{ url_for('static',
filename='article_generator.css') }}{% endblock %}

{% block header %}Article generator{% endblock %}

{% block content %}
<div class="article-container">
    <textarea id="user-input" placeholder="Enter your prompt
here..."></textarea>

```

```

    <div class="button-container">
      <button onclick="start_generating()" id="send-
button">Send</button>
      <button onclick="stop_generating()" id="stop-button"
style="background: grey" disabled>Stop</button>
      <label for="temp">Temperature:</label>
      <input type="number" id="temp" value="1" min="0.1"
max="5" step="0.1">
      <label for="top_p">Top_p:</label>
      <input type="number" id="top_p" value="1" min="0.1"
max="5" step="0.1">
    </div>
  </div>

  <script
src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.3.2/so
cket.io.js"></script>
  <script src="{{ url_for('static',
filename='article_generator.js') }}"></script>
{% endblock %}

```

## article\_generator.css

```

.article-container {
  max-width: 1000px;
  margin: 50px auto;
  padding: 20px;
  background-color: #fff;
  border-radius: 8px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

textarea {
  resize: none;
  box-sizing: border-box;
  width: 100%;
  min-height: 200px;
  margin-bottom: 20px;
  padding: 10px;
  border: 1px solid #ddd;
}

```



```

    border-radius: 5px;
}

.button-container {
    display: inline-block;
}

button {
    padding: 8px 20px;
    background-color: #007bff;
    color: #fff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    margin-right: 10px;
}

button:hover {
    background-color: #0056b3;
}

```

#### article\_generator.js

```

// create a socket with the server in order to communicate
var socket = io.connect('ws://' + document.domain + ':' +
location.port);

var textArea = document.getElementById('user-input');
var sendButton = document.getElementById('send-button');
var stopButton = document.getElementById('stop-button');
var tempInput = document.getElementById('temp')
var topPInput = document.getElementById('top_p')

isGenerating = false;

function start_generating() {
    /**
     Called when the send button is pressed. Sends the server the
     prompt and inference settings.
    */
}

```

```
var input = textArea.value;
var temperature = tempInput.value;
var top_p_val = topPInput.value;

onGenerationStart();

socket.emit('generate_article', {prompt: input, temp:
temperature, top_p: top_p_val});
}

function stop_generating() {
  /**
   Called when the stop button is pressed, tells the server to
stop generating a response.
  */
  if (!isGenerating)
    return;

  onGenerationEnd();
  socket.emit('stop_generating', {user_id: 'dang'});
}

socket.on('response', function(data) {
  /**
   Handles responses from the server. Every token generated and
sent by the server will call this event.
  */
  if (data['token'] == '</s>') {
    onGenerationEnd();
    return;
  }

  // add the new token to the text box
  textArea.value += data['token'];
});

function onGenerationStart() {
  /**
   Updates the send/stop buttons when the model start
generating a response.
  */
  isGenerating = true;
```

```

    stopButton.disabled = false;
    stopButton.style.background = '#007bff';
    sendButton.disabled = true;
}

function onGenerationEnd() {
    /**
     * Updates the send/stop buttons when the model stops
     * generating a response.
     */
    isGenerating = false;

    stopButton.disabled = true;
    stopButton.style.background = 'grey';
    sendButton.disabled = false;
}

```

compute\_server.html

```

{% extends 'base.html' %}

{% block stylesheet %}{{ url_for('static',
filename='compute_server.css') }}{% endblock %}

{% block header %}Compute server configuration{% endblock %}

{% block content %}
<div class="servers-container">
  <h1>Compute Servers</h1>
  <table>
    <thead>
      <tr>
        <th>Machine Name</th>
        <th>IP Address</th>
        <th>Port</th>
        <th>Model Name</th>
        <th>Compute device</th>
      </tr>
    </thead>
    <tbody>

```

```

    {% for server in server_list %}
    <tr>
      <td>{{ server.machine_name }}</td>
      <td>{{ server.ip }}</td>
      <td>{{ server.port }}</td>
      <td>{{ server.model_name }}</td>
      <td>{{ server.tensorflow_device }}</td>
    </tr>
    {% endfor %}
  </tbody>
</table>
</div>
{% endblock %}

```

## compute\_server.css

```

.servers-container {
  max-width: 1000px;
  margin: 50px auto;
  padding: 20px;
  background-color: #fff;
  border-radius: 8px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

table {
  width: 100%;
  border-collapse: collapse;
}

th {
  padding: 8px;
  text-align: left;
  border-bottom: 1px solid #ddd;
  background-color: #f2f2f2;
}

td {
  padding: 8px;
  text-align: left;
  border-bottom: 1px solid #ddd;
}

```

```
}
```

## מנהל בקשות

Requests.py

```
import abc
from enum import Enum

class RequestType(Enum):
    """
    Contains all request types in enum form.
    """
    NONE = -1
    QUESTION = 0
    ARTICLE = 1

class Request:
    """
    An abstract request class.

    Attributes:
        request_type: an enum containing the request type.
        token_handler: a reference to a function which handles
        generated tokens.
        active: whether the request is still active.
        waiting_for_response: whether the request is waiting for a
        response from a compute server.
    """
    def __init__(self, request_type):
        self.request_type = request_type
        self.token_handler = None
        self.active = True
        self.waiting_for_response = False

    @abc.abstractmethod
    def get_compute_data(self):
        """
        Gets the necessary data for sending a compute request.

```

```

        :return: the prompt along with inference parameters.
        """

@abc.abstractmethod
def handle_compute_result(self, token):
    """
    Called when the compute server returns a result.

    :param token: the token generated by the computer server.
    """

@abc.abstractmethod
def set_token_handler(self, token_handler):
    """
    Sets the request's token handler. This method will be
    called every time a new token is generated for the request.

    :param token_handler: a reference to the callback method.
    """

class QuestionRequest(Request):
    """
    Implementation of a question request, which gets sent to the
    chatbot.

    Attributes:
        prompt: contains the current generated article, sent to
        the compute server to generate the next token.
        inference_config: the inference configuration for the
        request, temperature and top_p.
    """

    def __init__(self, prompt, temp, top_p):
        """
        Create a question request object.

        :param prompt: the user's question.
        :param temp: the sampler's temperature.
        :param top_p: the sampler's top_p value.
        """

```

```

    super().__init__(RequestType.QUESTION)
    self.content = f'Below is an instruction that describes a
task. Write a response that appropriately completes the
request.\n\n### Instruction:\n{prompt}\n\n### Response:\n'
    self.inference_config = (temp, top_p)

def get_compute_data(self):
    """
    Gets the compute tasks's parameters.

    :return: the prompt and inference config for the compute
task.
    """
    self.waiting_for_response = True

    return self.content, self.inference_config

def handle_compute_result(self, token):
    """
    Handles a compute server result. Calls the request's token
handler.

    :param token: the token generated by the computer server.
    """
    self.waiting_for_response = False

    if token == '</s>':
        self.active = False
    else:
        self.content += token

    if self.token_handler is not None:
        self.token_handler(token)

def set_token_handler(self, token_handler):
    """
    Adds a token handler to the request. The token handler
will get called every time a new token is generated for the
response.

    :param token_handler: a reference to the token handler
method.
    """

```

```

        self.token_handler = token_handler

class ArticleRequest(Request):
    """
    Implementation of an article request.

    Attributes:
        prompt: contains the current generated article, sent to
        the compute server to generate the next token.
        inference_config: the inference configuration for the
        request, temperature and top_p.
    """
    def __init__(self, prompt, temp, top_p):
        """
        Create an article request object.

        :param prompt: the starting prompt by the user, can be
        empty.
        :param temp: the sampler's temperature.
        :param top_p: the sampler's top_p value.
        """
        super().__init__(RequestType.ARTICLE)
        self.content = prompt
        self.inference_config = (temp, top_p)

    def get_compute_data(self):
        """
        Gets the compute tasks's parameters.

        :return: the prompt and inference config for the compute
        task.
        """
        self.waiting_for_response = True

        return self.content, self.inference_config

    def handle_compute_result(self, token):
        """
        Handles a compute server result. Calls the request's token
        handler.

```



```

:param token: the token generated by the computer server.
"""
self.waiting_for_response = False

# if and end-of-text token has been generated, this
request is done and therefore no longer active
if token == '</s>':
    self.active = False
else:
    self.content += token

if self.token_handler is not None:
    self.token_handler(token)

def set_token_handler(self, token_handler):
    """
    Adds a token handler to the request. The token handler
    will get called every time a new token is generated for the
    response.
    :param token_handler: a reference to the token handler
    method.
    """
    self.token_handler = token_handler

```

## RequestManager.py

```

import time

class RequestManager:
    """
    Connects the web server and compute server by managing
    requests. The web server creates requests and adds them to the
    request manager. The request manager in turn sorts the them by
    priority and dispatches compute tasks when resources are
    available.

    Attributes:
        compute_server_manager: a reference to the compute server
        manager, used for dispatching compute tasks.
    """

```

```

    active_requests: a dictionary containing all requests, the
    user's id acts as the key.
    queue: contains the requests sorted by their priority in
    the form of a queue.
    """

    def __init__(self, compute_server_manager):
        self.compute_server_manager = compute_server_manager
        self.active_requests = dict()
        self.queue = list()

    def add_request(self, sid, request):
        """
        Adds a request to the request manager.

        :param sid: the id of the user who added the request.
        :param request: the request object itself
        """
        if sid in self.active_requests:
            raise Exception('TODO: implement sid collision logic')

        self.active_requests[sid] = request
        self.queue.append(request)

    def remove_request(self, sid):
        """
        Removes a request from the request manager.

        :param sid: the id of the request's user
        """
        if sid not in self.active_requests:
            raise Exception(f'Tried to remove request but no
            matching sid was found sid={sid}')

        request = self.active_requests.pop(sid)
        self.queue.remove(request)

    def get_request_for_server(self, server):
        """
        Finds a request which can be handled by a given compute
        server. Request with higher priority in the queue will be
        tended to first.

```

```

        :param server: the compute server.
        :return: a reference to the request object, if not found
        None will be returned.
        """
        for request in self.queue:
            if not request.waiting_for_response and
request.request_type == server.request_type:
                return request

        return None

    def dispatch_requests(self):
        """
        Continuously checks for available compute servers and
        dispatches compute requests. This method runs in its own
        thread.
        """
        while True:
            compute_servers =
self.compute_server_manager.get_available_servers()
            for server in compute_servers:
                request = self.get_request_for_server(server)
                if request is None:
                    continue

                # move request to the back of the queue
                self.queue.remove(request)
                self.queue.append(request)

                request_data = request.get_compute_data()
                callback = request.handle_compute_result
                server.compute(request_data, callback)

            time.sleep(0.01)

```

שרת מחשוב

ComputeMachine.py

```
from Requests import RequestType
```

```

import threading
from shared.Packets import *

class ComputeMachine(threading.Thread):
    """
    Represents a compute machine. Handles the traffic between
    the compute machine and the compute server manager. Acts as an
    interface for the request manager to dispatch compute tasks.

    Attributes:
        socket: the socket used to communicate with the compute
        server.
        ip: the ip address of the compute server.
        port: the port of the compute server.
        compute_server_manager: a reference to the compute server
        manager object.
        available: whether the compute server can handle new
        compute tasks.
        lock: a thread lock object to prevent race conditions.
        result_callback: a callback for the current compute task
        result.
        request_type: the request type this compute server can
        handle.
        machine_name: the name of the compute server.
        model_name: the name of the model the compute server is
        running.
        tensorflow_device: the device the compute server is
        running the model on.
    """

    def __init__(self, socket, computer_server_manager):
        """
        Creates a compute server object.

        :param socket: the socket for communicating with the
        compute server.
        :param computer_server_manager: a reference to the compute
        server manager.
        """
        super().__init__()
        self.socket = socket

```

```

self.ip, self.port = socket.getpeername()

self.computer_server_manager = computer_server_manager

self.available = True

self.lock = threading.Lock()
self.result_callback = None

self.request_type = RequestType.NONE

self.machine_name = 'N/A'
self.model_name = 'N/A'
self.tensorflow_device = 'N/A'

def run(self):
    """
    Listens for traffic from the compute server. This method
    runs in a separate thread.
    """
    try:
        while True:
            # wait for data from the compute server
            data = self.socket.recv(1024)

            if not data:
                print('Closed connection with:',
self.socket.getpeername())
                self.computer_server_manager.remove_server(self)
                return

            packet = decode_packet(data)

            # handle the packet
            if isinstance(packet, CPacketComputeResult):
                with self.lock:
                    if self.result_callback is None:
                        raise Exception(f'Result callback not found for
result {packet.result}')

                    self.result_callback(packet.result)
                    self.result_callback = None

```

```

        self.available = True
    elif isinstance(packet, CPacketMachineInfo):
        with self.lock:
            self.machine_name = packet.machine_name

            self.model_name = packet.model_name
            if self.model_name == 'article_10m':
                self.request_type = RequestType.ARTICLE
            elif self.model_name == 'chatbot_125m':
                self.request_type = RequestType.QUESTION
            else:
                raise Exception(f'Unknown model name
model_name={self.model_name}')

            self.tensorflow_device = packet.tensorflow_device
        else:
            print(f'No handle logic for packet: {packet}')

    except ConnectionError:
        print('Closed connection with:',
self.socket.getpeername())
        self.computer_server_manager.remove_server(self)

    def compute(self, request_data, callback):
        """
        Sends a compute task to the compute server.

        :param request_data: the compute task data, prompt and
inference config.
        :param callback: a reference to the result callback
method.
        """
        with self.lock:
            if not self.available:
                raise Exception('Tried computing with an unavailable
machine!')

            self.available = False
            self.result_callback = callback

            prompt, (temp, top_p) = request_data

```

```
packet = SPacketComputeTask(prompt, temp, top_p)
self.socket.send(encode_packet(packet))
```

### ComputeServerManager.py

```
import socket
import threading
from ComputeMachine import ComputeMachine

class ComputeServerManager(threading.Thread):
    """
    Manages active compute servers and handles new connections.

    Attributes:
        host: the compute server manager's ip.
        port: the port on which the compute server manager will
        listen for new connections.
        lock: a thread lock object to prevent race conditions.
        compute_machines: a list containing all the active compute
        servers.
    """

    def __init__(self, host, port):
        """
        Create the computer server manager.

        :param host: the compute server manager's host ip.
        :param port: the port on which the compute server manager
        will listen for new connections.
        """
        super().__init__()

        self.host = host
        self.port = port

        self.lock = threading.Lock()
        self.compute_machines = list()

    def get_available_servers(self):
        """
```

Returns a list of compute servers which are not handling a compute task.

:return: a list of compute servers that can handle new compute tasks.

```

"""
res = list()

for s in self.compute_machines:
    if s.available:
        res.append(s)

return res

def remove_server(self, server):
    """
    Removes a compute server from the compute server manager.
    :param server: the server to remove.
    """
    if server not in self.compute_machines:
        raise Exception(f'Tried removing server which is not in
the server list. server={server}')

    self.compute_machines.remove(server)

def run(self):
    """
    Listens for new connections from compute servers and adds
    them to the manager. This method runs in a different thread.
    """

    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind((self.host, self.port))
    s.listen(5)

    print('Compute server listening on port:', self.port)

    while True:
        # wait for a new connection
        client_socket, address = s.accept()

        with self.lock:

```



```

        compute_machine = ComputeMachine(client_socket, self)
        compute_machine.start()
        self.compute_machines.append(compute_machine)

        print('New compute machine added with address:',
address)

    s.close()

```

Packets.py

```

import json
import abc

class Packet:
    """
    An abstract class of a packet.
    """

    @abc.abstractmethod
    def to_json(self):
        """
        Serializes the packet to json form.
        :return: the json representation of the packet.
        """

    @staticmethod
    @abc.abstractmethod
    def from_json(json_str):
        """
        Deserialized the packet from json form.

        :param json_str: the packet in json form.
        :return: the deserialized packet object.
        """

# ----- Client-bound packets -----

```

```

class SPacketSetModel(Packet):
    """
    Tells the compute server to change its current model.

    !!! Note: the logic for this packet has yet to be
    implemented !!!

    Attributes:
        model_name: the name of the new model.
    """
    def __init__(self, model_name):
        self.model_name = model_name

    def to_json(self):
        return json.dumps({'model_name': self.model_name})

    @staticmethod
    def from_json(json_str):
        parsed = json.loads(json_str)
        return SPacketSetModel(parsed['model_name'])

class SPacketComputeTask:
    """
    Sends a compute task to the compute server.

    Attributes:
        prompt: the task prompt.
        temp: the sampler's temperature.
        top_p the sampler's top_p value.
    """
    def __init__(self, prompt, temp, top_p):
        if not isinstance(temp, float) or not isinstance(top_p,
float):
            raise Exception('Tried creating a SPacketComputeTask
with temperature or top_p that is not a floating point
number')

        self.prompt = prompt
        self.temp = temp
        self.top_p = top_p

```

```

def to_json(self):
    return json.dumps({'prompt': self.prompt, 'temp':
self.temp, 'top_p': self.top_p})

    @staticmethod
    def from_json(json_str):
        parsed = json.loads(json_str)
        return SPacketComputeTask(parsed['prompt'],
parsed['temp'], parsed['top_p'])

# ----- Server-bound packets -----

class CPacketComputeResult:
    """
    Contains a compute task's result.

    Attributes:
        result: the result of the compute task.
    """
    def __init__(self, result):
        self.result = result

    def to_json(self):
        return json.dumps({'result': self.result})

    @staticmethod
    def from_json(json_str):
        parsed = json.loads(json_str)
        return CPacketComputeResult(parsed['result'])

class CPacketMachineInfo:
    """
    Tells the compute server manager basic information about the
    compute server.

    Attributes:
        machine_name: the name of the compute server.
        model_name: the name of the model currently running on the
    compute server.

```

```

        tensorflow_device: the device being used to run the model.
    """
    def __init__(self, machine_name, model_name,
tensorflow_device):
        self.machine_name = machine_name
        self.model_name = model_name
        self.tensorflow_device = tensorflow_device

    def to_json(self):
        return json.dumps({'machine_name': self.machine_name,
            'model_name': self.model_name,
            'tensorflow_device': self.tensorflow_device})

    @staticmethod
    def from_json(json_str):
        parsed = json.loads(json_str)
        return CPacketMachineInfo(parsed['machine_name'],
parsed['model_name'], parsed['tensorflow_device'])

packet_registry = {
    0: SPacketSetModel,
    1: SPacketComputeTask,
    2: CPacketComputeResult,
    3: CPacketMachineInfo
}

def get_packet_by_id(packet_id):
    """
    Find a packet using it's registry ID.

    :param packet_id: the packet's ID.
    :return: the class of the packet corresponding to the given
ID.
    """
    if packet_id not in packet_registry:
        raise Exception(f'Tried handling a packet with an invalid
id! Got ID={packet_id}')

    return packet_registry[packet_id]

```

```
def get_id_from_packet(packet):
    """
    Gets the ID from a packet object.

    :param packet: the packet object.
    :return: the ID corresponding to the packet object.
    """
    for packet_id, packet_class in packet_registry.items():
        if isinstance(packet, packet_class):
            return packet_id

    raise Exception(f'Tried resolving the ID for a packet which
does not exist! Got packet: {packet}')

def encode_packet(packet):
    """
    Encodes the packet in order for it to be sent. Contains the
    ID and serialized packet.

    :param packet: the packet to encode.
    :return: the encoded packet in utf-8 form.
    """
    packet_id = get_id_from_packet(packet)
    serialized = packet.to_json()

    return bytes([packet_id]) + serialized.encode('utf-8')

def decode_packet(data):
    """
    Decodes an encoded packet. The encoded data contains the id
    and serialized packet.

    :param data: the encoded data.
    :return: a packet object from the encoded data.
    """
    packet_id = data[0]
    serialized = data[1:]

    return get_packet_by_id(packet_id).from_json(serialized)
```

## ComputeClient.py

```
import sys
import socket
from Packets import *

def main(host, port, machine_name, model_name):
    """
    Initializes the compute server and maintains connections
    with the main compute server.

    :param host: the address of the main compute server.
    :param port: the port of the main compute server.
    :param machine_name: the name of this compute server.
    :param model_name: the name of the model to run on this
    compute server.
    """

    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((host, port))

    # models are imported separately as they need different
    libraries
    model = None
    if model_name == 'article_10m':
        from ArticleModel import ArticleModel
        model = ArticleModel()
    elif model_name == 'chatbot_125m':
        from ChatbotModel import ChatbotModel
        model = ChatbotModel()
    else:
        raise Exception(f'Invalid model name,
model_name={model_name}')

    model.load_model()
    model.load_tokenizer()

    tensorflow_device = model.get_tensorflow_device()
```

```

    info_packet = CPacketMachineInfo(machine_name, model_name,
    tensorflow_device)
    s.send(encode_packet(info_packet))

    # main compute loop
    while True:
        data = s.recv(65536)
        packet = decode_packet(data)

        result = model.compute(packet.prompt, packet.temp,
    packet.top_p)
        print(f'Generated token with temp={packet.temp}
    top_p={packet.top_p} => {result}')

        response = CPacketComputeResult(result)
        s.send(encode_packet(response))

if __name__ == '__main__':
    """
    Script entry point.

    The script needs to get the following arguments:
    1. address of main compute server
    2. port of main compute server
    3. name of this compute server
    4. name of the model to run
    """
    args = sys.argv[1:]

    if len(args) != 4:
        raise Exception('Invalid number of script arguments! Must
    be 4.')

    args[1] = int(args[1]) # convert port to integer

    main(*args)

```

ArticleModel.py

```

import tensorflow as tf
import numpy as np
from Transformer import TransformerBlock,
Multi_Head_Attention, FeedForwardNetwork

```

```

from PositionalEmbedding import PositionalEmbedding
from Tokenizer import Tokenizer

CHECKPOINT_DIR = '/checkpoints/article_checkpoint.ckpt'
TOKENIZER_DIR = '/tokenizers/tokenizer_2500_vocab.json'

PAD_TOKEN = '<pad>'
START_TOKEN = '<s>'
END_TOKEN = '</s>'

# model hyperparameters
BATCH_SIZE = 64
BUFFER_SIZE = 50_000
train_split = 0.8
validation_split = 0.1
test_split = 0.1
vocab_size = 2500
transformer_units = 6
d_model = 384
seq_length = 256
attention_heads = 8
dff = 1024 # transformer ffn hidden layer size
dropout_rate = 0.1
learning_rate = 1e-6

class ArticleModel:
    """
    Class abstracting the 'article_10m' model.

    Attributes:
        model: the model object loaded into memory.
        tokenizer: the model's tokenizer.
        model_name: the name of the model.
    """

    def __init__(self):
        self.model = None
        self.tokenizer = None
        self.model_name = 'article_10m'

```



```

@staticmethod
def get_tensorflow_device():
    """
    Gets the devices tensorflow is using to run the model.

    :return: A list of all devices the model can run on.
    """
    return tf.config.list_physical_devices()

def load_model(self):
    """
    Loads the model checkpoint.

    Because the checkpoint only contains the weights of the
    model, it first needs to be built.
    """
    inputs = tf.keras.Input(shape=(seq_length,))
    # token embedding layer
    x = tf.keras.layers.Embedding(vocab_size, d_model,
mask_zero=True)(inputs)
    # positional embedding layer
    x = PositionalEmbedding(5000, d_model)(x)

    # stack transformers on top of each other
    for _ in range(transformer_units):
        x = TransformerBlock(attention_heads, dff, d_model,
dropout_rate)(x)

    x = tf.keras.layers.LayerNormalization()(x)

    outputs = tf.keras.layers.Dense(vocab_size)(x)

    model = tf.keras.Model(inputs=inputs, outputs=outputs)

    model.compile(loss=tf.keras.losses.SparseCategoricalCrosse
ntropy(from_logits=True),
                optimizer=tf.keras.optimizers.legacy.Adam(learning_
rate=learning_rate),
                metrics=['accuracy'])

    model.load_weights(CHECKPOINT_DIR)

```

```

self.model = model

def load_tokenizer(self):
    """
    Loads the pretrained tokenizer.
    """
    tokenizer = Tokenizer(2500)
    tokenizer.deserialize_from_file(TOKENIZER_DIR)

    self.tokenizer = tokenizer

def compute(self, prompt, temp=0.5, top_p=0.5):
    """
    Generates a single token given a context.

    :param prompt: the context used to generate to token.
    :param temp: the sampler's temperature.
    :param top_p: the sampler's top_p value.
    :return: a single token sampled from the model's output
distribution.
    """
    if self.model is None:
        raise Exception('Tried computing with a model that
hasn\'t been loaded yet!')

    if self.tokenizer is None:
        raise Exception('Tried using a tokenizer that hasn\'t
been loaded yet!')

    tokenized = [2] + self.tokenizer.tokenize(prompt)
    tokenized = tokenized[-256:] # use the last 256 tokens
    index = len(tokenized) - 1
    window = tokenized + [0] * (seq_length - len(tokenized))

    pred = self.model(np.array([window]))
    pred = np.squeeze(pred)[index].astype('float64')
    pred /= float(temp)

    pred = self.top_p_function(pred, top_p)

    # sample a token using the multinomial function
    token = np.argmax(np.random.multinomial(1, pred))

```

```

        decoded_token = self.tokenizer.detokenize([token])

        return decoded_token

    @staticmethod
    def top_p_function(logits, threshold):
        """
        Applies the top_p function to a vector of logits. Returns
        the normalized probability of the most probable token's whos
        combined probability doesn't exceed threshold.

        :param logits: the logits to apply the top_p function to.
        :param threshold: the top_p value.
        :return: the probability distribution of the logits after
        applying the top_p function.
        """
        # type must be float64 due to softmax function rounding
        errors
        logits = tf.cast(logits, dtype=tf.float64)
        probs = tf.nn.softmax(logits)

        # sort and calculate cumulative sums
        sorted_probs, sorted_indices = tf.math.top_k(probs,
            k=tf.shape(probs)[-1], sorted=True)
        cum_probs = tf.cumsum(sorted_probs, axis=-1,
            exclusive=True)

        # mask out tokens outside the top_p threshold
        mask = tf.cast(cum_probs <= threshold, dtype=tf.float64)
        masked_probs = sorted_probs * mask
        renorm_probs = masked_probs / tf.reduce_sum(masked_probs,
            axis=-1, keepdims=True)

        # create new probabilities
        new_probs =
            tf.scatter_nd(indices=tf.expand_dims(sorted_indices, axis=-1),
                updates=renorm_probs,
                shape=tf.shape(probs))

        return new_probs

```

ChatbotModel.py

```
import os
```

```

import keras_nlp
import keras
import tensorflow as tf
import numpy as np

os.environ["KERAS_BACKEND"] = "tensorflow"
keras.mixed_precision.set_global_policy("mixed_float16")

seq_length = 2048

class ChatbotModel:
    """
    Class abstracting the 'article_10m' model.

    Attributes:
        model: the model object loaded into memory.
        tokenizer: the model's tokenizer.
        model_name: the name of the model.
    """

    def __init__(self):
        self.model = None
        self.tokenizer = None
        self.model_name = 'chatbot_125m'

    @staticmethod
    def get_tensorflow_device():
        """
        Gets the devices tensorflow is using to run the model.

        :return: A list of all devices the model can run on.
        """
        return tf.config.list_physical_devices()

    def load_model(self):
        """
        Loads the model checkpoint.

        The checkpoint only contains weights, and therefore the
        model needs to be built beforehand. In this case the model is
        downloaded.

```

```

        """
        self.model =
keras_nlp.models.OPTCausalLM.from_preset("opt_125m_en",
preprocessor=None)
        self.model.compile(loss=tf.keras.losses.SparseCategoricalC
rossentropy(from_logits=True),
                        optimizer=tf.keras.optimizers.Adam(learning_rate
=1e-4),
                        metrics=['accuracy'])

        self.model.load_weights('checkpoint.weights.h5')

def load_tokenizer(self):
    """
    Loads the pretrained tokenizer.
    """
    self.tokenizer =
keras_nlp.models.OPTTokenizer.from_preset("opt_125m_en")

def compute(self, prompt, temp=0.5, top_p=0.5):
    """
    Generates a single token given a context.

    :param prompt: the context used to generate to token.
    :param temp: the sampler's temperature.
    :param top_p: the sampler's top_p value.
    :return: a single token sampled from the model's output
distribution.
    """
    if self.model is None:
        raise Exception('Tried computing with a model that
hasn\'t been loaded yet!')

    if self.tokenizer is None:
        raise Exception('Tried using a tokenizer that hasn\'t
been loaded yet!')

    window = [0] +
tokenizer(prompt).numpy().tolist()[seq_length - 1]

    padding_mask = [1] * len(window)

```

```

model_input = {
    'token_ids': np.array([window]),
    'padding_mask': np.array([padding_mask])
}

pred = model(model_input, training=False)
pred = tf.squeeze(pred[0, -1, :]) # only consider next
position predictions
pred /= temp

pred = top_p_function(pred, top_p)

token = np.argmax(np.random.multinomial(1, pred))
decoded_token =
tokenizer.detokenize([token]).numpy().decode("utf-8")

return decoded_token

@staticmethod
def top_p_function(logits, threshold):
    """
    Applies the top_p function to a vector of logits. Returns
    the normalized probability of the most probable token's whos
    combined probability doesn't exceed threshold.

    :param logits: the logits to apply the top_p function to.
    :param threshold: the top_p value.
    :return: the probability distribution of the logits after
    applying the top_p function.
    """
    # type must be float64 due to softmax function rounding
    errors
    logits = tf.cast(logits, dtype=tf.float64)
    probs = tf.nn.softmax(logits)

    # sort and calculate cumulative sums
    sorted_probs, sorted_indices = tf.math.top_k(probs,
k=tf.shape(probs)[-1], sorted=True)
    cum_probs = tf.cumsum(sorted_probs, axis=-1,
exclusive=True)

    # mask out tokens outside the top_p threshold

```

```

    mask = tf.cast(cum_probs <= threshold, dtype=tf.float64)
    masked_probs = sorted_probs * mask
    renorm_probs = masked_probs / tf.reduce_sum(masked_probs,
axis=-1, keepdims=True)

    # create new probabilities
    new_probs =
tf.scatter_nd(indices=tf.expand_dims(sorted_indices, axis=-1),
              updates=renorm_probs,
              shape=tf.shape(probs))
    return new_probs

```

Tokenizer.py

קובץ זה מכיל את מחלקת tokenizer של המודל הראשון. תוכן הקובץ זהה למחלקה שנכתבה בקובץ "Transformer\_2500\_vocab.ipynb" (עמוד 76). לכן, לא אצרף אותו שוב.

Transformer.py

קובץ זה מכיל היישומים של מחלקות הטרנספורמר: FeedForwardNetwork, MultiHeadAttention וTransformerBlock. תוכן הקובץ זהה ליישום המחלקות שמופיעים בקובץ "Transformer\_2500\_vocab.ipynb" (עמוד 92). לכן, לא אצרף אותו שוב.

PositionalEmbedding.py

קובץ זה מכיל את היישום של שכבת קידוד המיקום. תוכן הקובץ זהה ליישום שמופיע בקובץ "Transformer\_2500\_vocab.ipynb" (עמוד 94). לכן, לא ארצף אותו שוב.

## מדריך למשתמש

בעקבות האופי האזוטרי והמסובך של היישום, ישנם שתי דרכים להשתמש במודלים. דרך אחת היא באמצעות היישום המקורי הדורש מאמץ רב. הדרך השנייה היא באמצעות סביבת הפיתוח בענן Google Colab. בגלל מגבלות הסביבה לא ניתן להוסיף את הממשקים שקיימים ביישום הרגיל, לכן דרך זו נועדה רק לתקשורת ישירה וקלה עם המודלים, ללא ממשק.

### דרך סביבת פיתוח Google Colab

השלבים להרצת המודל הראשון

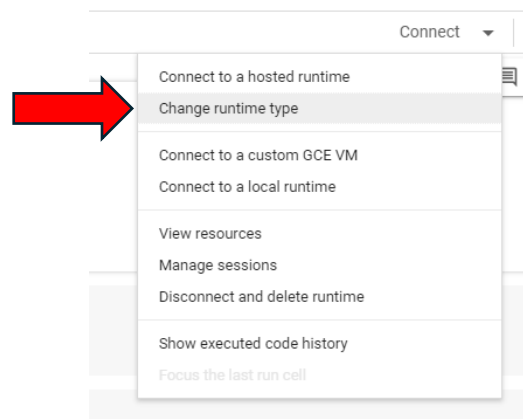
1. יש להיכנס לקישור הבא אשר פותח את הפרויקט בסביבת Google Colab:  
<https://colab.research.google.com/drive/1u18clYWDvUwG73hnYY7zmpeXeswVIOzZ?usp=sharing>

2. בכניסה לאתר יש למצוא את כפתור Connect אשר מחבר את המשתמש לסביבת ההרצה. הכפתור ממוקם בצד ימין של המסך.



מצידו הימין של הכפתור ישנו כפתור נוסף בצורת משולש הפוך, יש ללחוץ עליו כדי לפתוח את הגדרות הסביבה.





בתפריט שתפתח יש ללחוץ על "Change runtime type"

### Change runtime type

#### Runtime type

Python 3 ▼

#### Hardware accelerator ?

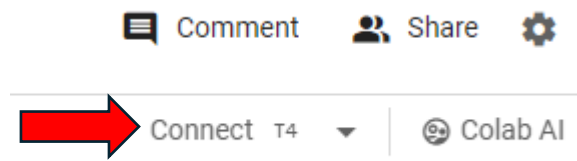
- ☐ CPU
 ☒ T4 GPU
 ☐ A100 GPU
 ☐ L4 GPU
- ☐ V100 GPU (deprecated)
 ☐ TPU (deprecated)
- ☐ TPU v2

Want access to premium GPUs? [Purchase additional compute units](#)

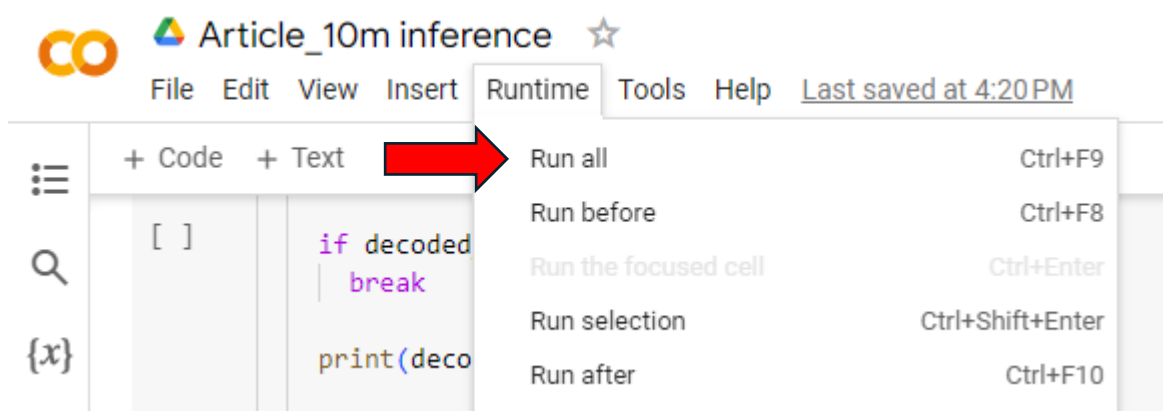
Cancel Save

לאחר מכן, יפתח חלון ובו הגדרות לסביבת ההרצה. יש לוודא כי תחת "Hardware accelerator" מסומנת האופציה "T4 GPU", וללחוץ על כפתור השמירה שממוקם בצד ימין למטה בחלון.

לבסוף יש ללחוץ על כפתור Connect כדי להתחבר ולהתחיל את סביבת ההרצה.



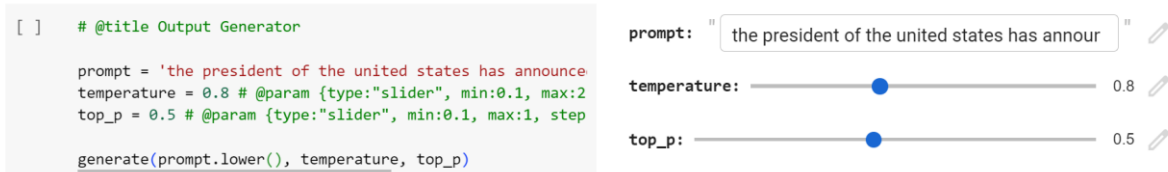
3. הפרויקט מכיל מספר פרקים שמורידים קבצים, מגדירים מחלקות, טוענים תצורות וכו'. הפרק שבו ניתן להשתמש במודל הוא הפרק האחרון המכונה "Inference". חובה להריץ את כל חלוני הקוד הנמצאים בפרויקט. ניתן לעשות זאת באמצעות לחיצה על תפריט Runtime ובחירת האופציה הראשונה "Run all"



4. כעת ניתן לגלול למטה ולמצוא את פרק Inference שבו חלון קוד יחיד שמפשט את הרצת המודל.

#### ▼ Inference

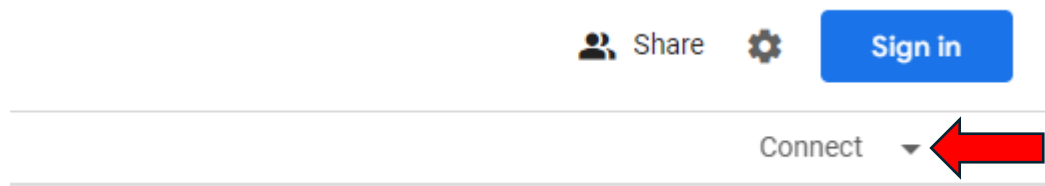
#### ▼ Output Generator



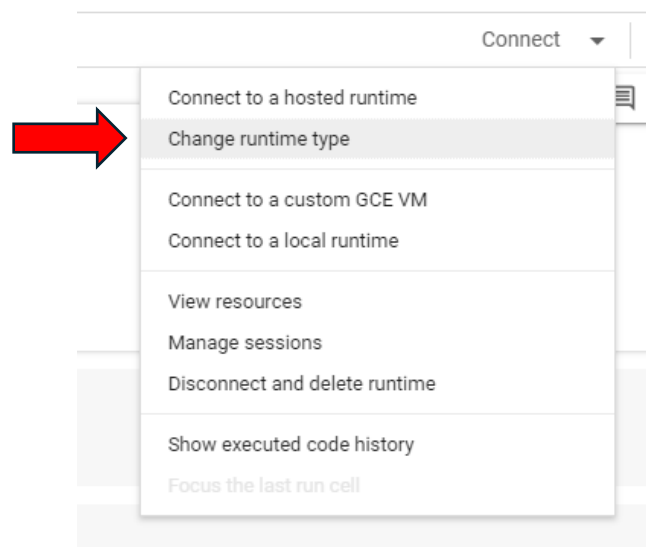
בצידו הימני של חלון הקוד ניתן למצוא הגדרות להרצת המודל. שדה prompt מכיל את תחילת הכתבה שהמודל יצור. שדות temperature ו-top\_p מכילות את הגדרות הסמפלר. יש לעיין בהסבר על הסמפלר בפרק מבנה הפרויקט, אולם אפשר לסכם ולומר כי ערכים גדולים גורמים ליותר אקריות. בהרצת הקוד, פלט המודל יופיע מתחת לחלון הקוד.

## השלבים להרצת המודל השני

1. יש להיכנס לקישור הבא אשר פותח את הפרויקט בסביבת Google Colab:  
<https://colab.research.google.com/drive/1ISMZXIMaVswlp9vO2mV10xv-0fWXQuNg?usp=sharing>
2. בכניסה לאתר יש למצוא את כפתור Connect אשר מחבר את המשתמש לסביבת ההרצה. הכפתור ממוקם בצד ימין של המסך.



מצידו הימין של הכפתור ישנו כפתור נוסף בצורת משולש הפוך, יש ללחוץ עליו כדי את הגדרות הסביבה.



בתפריט שתפתח יש ללחוץ על "Change runtime type"

Change runtime type

Runtime type

Python 3

Hardware accelerator ?

☐ CPU ☒ T4 GPU ☐ A100 GPU ☐ L4 GPU

☐ V100 GPU (deprecated) ☐ TPU (deprecated)

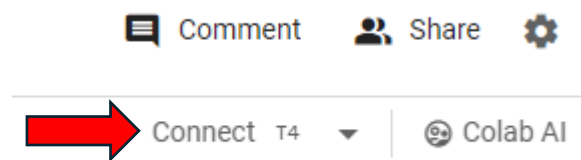
☐ TPU v2

Want access to premium GPUs? [Purchase additional compute units](#)

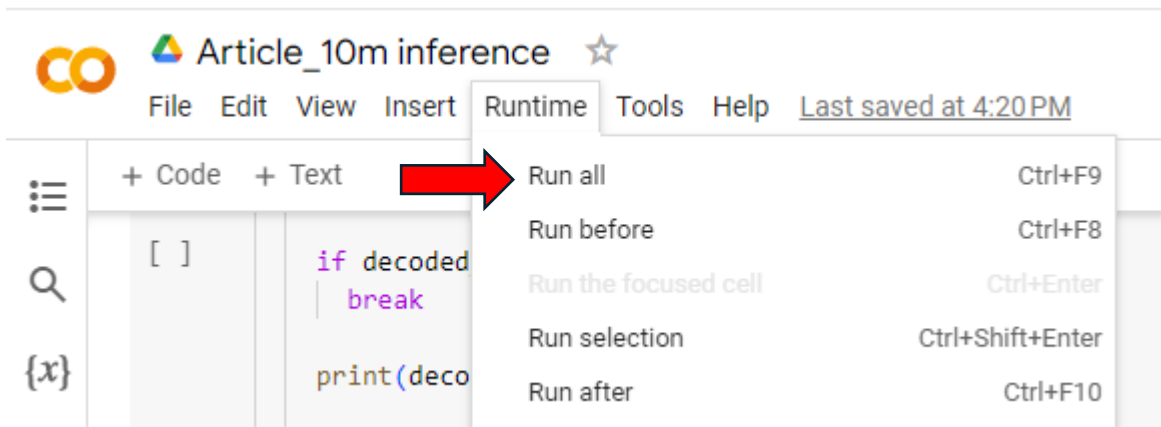
Cancel Save

לאחר מכן, יפתח חלון ובו הגדרות לסביבת ההרצה. יש לוודא כי תחת "Hardware accelerator" מסומנת האופציה "T4 GPU", וללחוץ על כפתור השמירה שממוקם בצד ימין למטה בחלון.

לבסוף יש ללחוץ על כפתור Connect כדי להתחבר ולהתחיל את סביבת ההרצה.



3. הפרויקט מכיל מספר פרקים שמורידים קבצים, טוענים תצורות וכו'. הפרק שבו ניתן להשתמש במודל הוא הפרק האחרון המכונה "Inference". חובה להריץ את כל חלוני הקוד הנמצאים בפרויקט. ניתן לעשות זאת באמצעות לחיצה על תפריט Runtime ובחירת האופציה הראשונה "Run all". בעקבות גודל המודל, הקבצים הינם כמה גיגה-ביטים, לכן שלב זה עשוי לקח מספר דקות.



5. כעת ניתן לגלול למטה ולמצוא את פרק Inference שבו חלון קוד יחיד שמפשט את הרצת המודל.

#### ▼ Output Generator

```

[13] # @title Output Generator

question = 'What is the cause of climate change?'
temperature = 0.5 # @param {type:"slider", min:0.1, max:1.0}
top_p = 0.5 # @param {type:"slider", min:0.1, max:1.0}
show_template = False # @param {type:"boolean"}

prompt = f'Below is an instruction that describes
generate(prompt, temperature, top_p, show_template

```

question: "What is the cause of climate change?"

temperature:  0.5

top\_p:  0.5

show\_template: ☐

The cause of climate change is the burning of fossil fuels, such as coal, oil, and natural gas. These fuels rel

בצידו הימני של חלון הקוד ניתן למצוא הגדרות להרצת המודל שאפשר לשנות. שדה question מכיל את השאלה שעליה ישיב המודל. שדות temperature ו-top\_p מכילות את הגדרות הסמפלר. יש לעיין בהסבר על הסמפלר בפרק מבנה הפרויקט, אולם אפשר לסכם ולומר כי ערכים גדולים גורמים ליותר אקריות. השדה show\_template מגדיר האם להציג את תבנית השאלה בנוסף לתשובה עצמה, כלומר הטקסט שהמודל רואה. בהרצת הקוד, פלט המודל יופיע מתחת לחלון הקוד.

## הרצת היישום המלא

### הורדה והרצת הפרויקט

היישום נועד להיות מורכב ממספר מחשבים שעובדים יחד, כאשר כל אחד אחראי על משימה ספציפית. למען פשטות, מדריך זה יסביר כיצד להריץ את כל חלקי היישום במחשב אחד. כדי להריץ את חלקי היישום במחשבים שונים יש להגדיר מספר משתנים הקובעים את אופי ניתוב תעבורת הרשת.

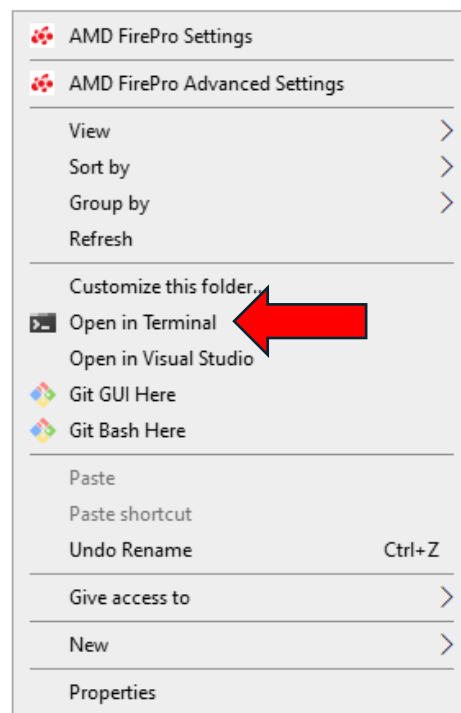
ישנם מספר מגבלות להרצת הפרויקט, רובם נובעים מהמודל השיחתי. אם רוצים להריץ את הפרויקט עם המודל הראשון בלבד, ניתן לעשות זאת במערכת הפעלה windows. אולם, כדי להריץ את המודל השיחתי, חובה להריץ את הפרויקט במערכת ההפעלה לינוקס. על כן, יש שני קבצים בתיקיית הפרויקט, אחד להרצת הפרויקט בסביבת windows ואחד להרצת הפרויקט בסביבת לינוקס.

1. יש להוריד את תיקיית הפרויקט מהקישור הבא:

[https://drive.google.com/drive/folders/1jhPMwYAa-7\\_zvfRHcP3tcQkX\\_uAOdCAw?usp=sharing](https://drive.google.com/drive/folders/1jhPMwYAa-7_zvfRHcP3tcQkX_uAOdCAw?usp=sharing)

2. ההנחיות ב2.1 הן להרצה בסביבת windows, ואילו ההנחיות ב2.2 הן להרצה בסביבת linux.

1.2 כדי להתחיל את היישום צריך להריץ את הקובץ "windows\_setup.py" שנמצא בתיקיית הפרויקט. ניתן לעשות זאת באמצעות הטרמינל, כדי לפתוח את הטרמינל בתיקייה הנוכחית יש ללחוץ לחיצה ימנית בתוך התיקייה הפתוחה ולבחור באופציה "Open in Terminal".



לאחר שהטרמינל נפתח, ניתן להריץ את הקובץ באמצעות הפקודה "python windows\_setup.py".

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Arzi0\OneDrive\Desktop\AI final project> python windows_setup.py
```

בטרמינל יודפסו עדכונים לגבי הורדת ספריות ואתחול קבצים. לפני שממשיכים לשלב הבא, יש לחכות עד שיתקבל פלט סיום ההפעלה: "setup complete".

1.2 כדי לאתחל את היישום בסביבת linux צריך להריץ את הקובץ "setup\_linux.py". יש לפתוח את הטרמינל, להיכנס דרכו לתיקיית הפרויקט ולהריץ את הקובץ באמצעות הפקודה "python stepup\_linux.py". בטרמינל יודפסו עדכונים לגבי הורדת ספריות ואתחול קבצים. לפני שממשיכים לשלב הבא, יש לחכות עד שיתקבל פלט סיום ההפעלה: "setup complete".

3. כעת, לאחר שהיישום אותחל, ניתן לגשת עליו דרך כל דפדפן בכתובת "localhost:5000". אפשר לגשת לאתר דרך המחשב בו היישום רץ בלבד.



## מבנה האתר

האתר מורכב ממספר עמודים שבכל אחד תוכן שונה. בחלק העליון של כל העמודים מופיע תפריט ניווט שמאפשרת מעבר בין העמודים השונים. לפני שניתן לגשת לעמודי התוכן, האתר יבקש הזדהות באמצעות שם משתמש וסיסמה. שם משתמש: admin סיסמה: iamtheadmin.

## עמוד אימות משתמש

- Please log in to access this page.

Username:

password:

Submit

כאשר מנסים להתחבר לאתר בפעם הראשונה, ממכשיר חדש או אחרי הרבה זמן תידרש אימות של המשתמש. זאת מכיוון שבשלב ההגנה על הפרויקט האתר יהיה פתוח לאינטרנט, ולכן צריך לאמת את המשתמש שמנסה להתחבר.

## עמוד הבית

**About the project**

Chatbot Article generator Compute server configuration About

**About this project**

The purpose of this website is to present two Natural Language Processing (NLP) neural networks that are part of a final project in an artificial intelligence course. Both models are powered by the transformer architecture found in other Generative Pretrained Transformer (GPT) models. In order to run these models in parallel and allow multiple users to run them at the same time, a network of computational servers was implemented. Each server hosts a model and is open to inference requests, which are dispatched by a central server.

The first model was trained to write news articles based on the input a user gives it. The input can be the first word of the article, first line, first paragraph or be left blank. Dataset wise, the model was trained on some 312,000 news articles sourced from CNN. It boasts 6 transformer decoder units stacked atop each other, as well as embedding, encoding and dense layers. The model has a total of 10 million trainable parameters, and was trained on an RTX 2060 for around 40 hours.

The second model is an upscaled version of the first model, containing 12 transformer blocks and higher dimensionality for a grand total of 125 million trainable parameters. Because of its size, training such a model from scratch is infeasible. Therefore, a pretrained model was obtained from the Open Pretrained Transformers project, which was trained on a general dataset. In order to fine tune the model for question answering a dataset was created by combining Stanford's LLaMa dataset with custom generated questions, for a total of 54,000 question-answer pairs. The model was trained for 8 hours on a T4 GPU provided by Google Colab.

עמוד זה מכיל הסבר מתומצת של הפרויקט והיישום. ניתן לחזור אליו בכל עת באמצעות לחיצה על כפתור הAbout בתפריט הניווט.



## עמוד המודל השיחתי

Chatbot interface

Chatbot Article generator Compute server configuration About

user  
What is your purpose?

chatbot  
My purpose is to educate people about sustainability.

Type your message... Send Stop

Temperature: 1 Top\_p: 1

עמוד זה מאפשר לתקשר ולדבר עם המודל השיחתי. הוא מכיל אזור צ'אט שבו יופיעו השאלות שנשלחו על ידי המשתמש והתשובות שהתקבלו מהמודל. מתחת לחלון הצ'אט נמצאת תיבת טקסט שבה ניתן לכתוב שאלות שישלחו למודל. מימינה יש שני כפתורים, כפתור שליחה אשר שולח את השאלה שנכתבה בתיבת הטקסט, וכפתור עצור אשר עוצר את המודל בעת כתיבת תשובה. לבסוף, יש שני שדות מספריים. שדות Temperature ו-Top\_p מגדירים את תצורת הסמפלר שקובע את מידת האקריות ויצירתיות של המודל. הסבר מלא על הערכים אלה נמצא בתת הפרק סמפלר שנצא בפרק מבנה הפרויקט (עמוד 46). פשר לסכם ולומר כי ערכים גדולים גורמים ליותר אקריות, וערכים קטנים עלולים לגרום לחזרה על אותם המילים או משפטים שוב ושוב.

## עמוד כתיבת כתבות

## Article generator

Chatbot
Article generator
Compute server configuration
About

Enter your prompt here...

Send
Stop
Temperature: 
Top\_p:

יצירת כתבות באמצעות המודל הראשון נעשה בעמוד זה. הוא מכיל תיבת טקסט גדולה שבה המשתמש יכול לכתוב את הprompt של המודל, אפשר לכתוב משפט שלם, כמה מילים ואף להשאיר את התיבה ריקה. מתחתיה יש שני כפתורים: כפתור שליחה, שגורם למודל להתחיל לכתוב, וכפתור עצור שעוצר את המודל בזמן כתיבה. שדות Temperature ו-Top\_p מגדירים את תצורת הסמפלר שקובע את מידת האקריות ויצירתיות של המודל. הסבר מלא על הערכים אלה נמצא בתת הפרק סמפלר שנצא בפרק מבנה הפרויקט (עמוד 46). אפשר לסכם ולומר כי ערכים גדולים גורמים ליותר אקריות, וערכים קטנים עלולים לגרום לחזרה על אותם המילים או משפטים שוב ושוב.

## עמוד הגדרות שרת המחשוב

Machine Name	IP Address	Port	Model Name	Compute device
linux_server	192.168.1.156	56690	chatbot_125m	[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]

עמוד זה מציג את הגדרות שרתי המחשוב. הוא מכיל טבלה שבה מופיעים כל שרתי המחשוב שמחוברים לשרת המחשוב המרכזי. הטבלה מורכבת מחמש שדות המציגות מידע על הגדרות השרת של הישות והמודל שרץ עליה. חשוב לציין כי אין אפשרות לשנות את השדות, הטבלה נועדה רק להציג מידע.

- Machine Name: שם שרת המחשוב
- IP Address: כתובת ה IP של השרת
- Port: מספר ה port של ה socket בצד שרת המחשב
- Model Name: שם המודל שהשרת מריץ
- Compute Device: המכשיר בו המודל רץ, מעבד או כרטיס מסך

## רפלקציה

הפרויקט הזה הוא אחד הדברים הקשים ביותר שעבדתי עליהם, אם לא הדבר הקשה ביותר שעבדתי עליו. למידת העקרונות של הבינה המלאכותית במשך שנה היה מעניין מאוד ויצר בסיס איתן ללמידה מתקדמת. אולם, הידע שנצבר בקורס לא מגרד את קצה הקרחון של הלמידה שנעשתה במסגרת הפרויקט. הוא מוכל בחמישה עמודים מתוך ארבעים בפרק מבנה הפרויקט. הידע שנמצא בשלושים וחמישה העמודים שבאים אחריו נרכש בפחות מחצי שנה.

התחלתי מרעיון מופרך, לבנות מודל שיחתי שמבוסס על אותה טכנולוגיה כמו המודלים שנמצאים בחזית המחקר הטכנולוגי. רציתי ללמוד כיצד המודלים הללו עובדים ולמה תוך שנים ספורות הם התקדמו במהירות חסרת תקדים. מאחר ומדובר בפרויקט שאצטרך להגיש בעתיד, החלטתי להתחיל ללמוד את המודל השפתי הפשוט ביותר, וכאשר ארגיש בקיאתחום אעבור למודל יותר מתקדם. כך בכל שלב של הפרויקט יהיה לי מודל שעליו אוכל להסתמך.

התחלתי מהמודל הפשוט ביותר, bigram וtrigram, שלמען האמת קשה לומר שהוא מבוסס על בינה מלאכותית. הוא מורכב מטבלה גדולה שמכילה את ההסתברויות שמילה תבוא אחרי מילה אחרת. לאחר מכן, למדתי כיצד ניתן לדמות את הטבלה הזו באמצעות קידוד מילים ורשת נוירונים. המשכתי בלמידה על מודל שפתי רקורסיבי RNN ואחיו LSTM. לבסוף, התחלתי ללמוד על מודלים שפתיים מבוססי ארכיטקטורת טרנספורמר.

חשבתי שלא אצליח להגיע לרמה של מודל עובד, ולימוד הטרנספורמר יהיה רק העשרה קטנה. אולם עם הזמן ועם בניית מודלים יותר ויותר משוכללים הרגשתי שאולי הטרנספורמר לא כה מפיח ומאיים כפי שחשבתי. לבסוף הכנתי מאגר נתונים בסיסי, בניתי מודל פשוט והשארתי אותו להתאמן בשמך כל הלילה. כשהתעוררתי ובדקתי את פלט המודל, ראיתי נצוץ של בינה שיחתית, שבין שורות המילים הכמעט אקדמיות, כאן ושם יש רצף מילים קוהרנטי שמנסה להביע רעיון.

לא ריציתי להסתפק בהבנה שטחית של המודל, אלא ממש לראות איך אותות המידע זורמים במנגנונים השונים. לכן, החלטתי לתכנת את השכבות השונות של מודל הטרנספורמר מאפס. אני לא יכול לספור את מספר הטעויות שהיו לי בקוד, טעויות שקשה להבחין בהן עד שמאמנים את המודל. הייתה טעות אחת במכפלת מטריצות שלקח לי יותר משבוע למצוא, באג שבא לידי ביטוי רק בסוף האימון כשמנסים להשתמש במודל.

לו היו אומרים לי כמה זמן אצטרך להשקיע בפרויקט, הייתי מצמצם את היקפו פי כמה וכמה. כשתכננתי את לוח הזמנים חשבתי שאצליח לסיים את הפרויקט בחמישית מהזמן. אך בכל שלב נתגלו בעיות שערכו אותי. גיליתי שנושאים, אשר לכאורה נראים פשוטים, לאמיתו של דבר דורשים ימים של מחקר. בעיות עם סביבות העבודה פקדו אותי ללא הרף, בין אם מדובר במחשוב ענן ובין אם מדובר במחשוב מקומי.

עם זאת, לא עשיתי את הפרויקט מתוך חובה, אלא מרצון. למרות שישבתי לילות כימים על מתמטיקה שאני לא מבין ובאגים עקשנים, נהייתי לאורך כל הדרך. הרגשתי שעם כל נושא חדש שנלמד, כל מודל ניסיוני שנשלח לאימון, אני מרכיב עוד חלק מפאזל ומדייק את ההבנה שלי של עולם הבינה המלאכותית.

למדתי על השכבות השונות של המודל. הבנתי את עוצמתה של שכבת הקידוד, את הצורך ב-tokenization וחשוב מכל איך הטרנספורמר עובד. במהלך קריאת מאמרים הבנתי מה כה מהפכני בארכיטקטורה הזו, וכיצד היא הצליחה לחולל מהפכה בעולם הבינה המלאכותית. רכשתי אינטואיציה לאיך נראה מודל יעיל, אילו שכבות צריכים ובאילו גדלים.

אני לוקח איתי ראייה חדשה של העולם, דרך עדשות הבינה המלאכותית. לדעת לזהות דפוסים במידע ואיך ניתן לבנות מודל שיכול להבין ולהסיק מסקנות מהקשרים הללו. התכונה המהפכנית של הבינה המלאכותית היא היכולת שלה להגיע למסקנות ולמצוא קשרים בין נתונים במהירות הבזק. מי שיוודע לרתום את היכולות האלה מחזיק בכוח אדיר. ובכדי לרתום את הבינה, צריך לדעת לפענח את המידע ולבנות מודלים מתאימים.

אי אפשר להסתפק רק בידע תאורטי, צריך לדעת גם איך ליישם את הרעיונות. בניית אתר האינטרנט, שרת המחשוב ומנהל הבקשות לימדו אותי כיצד להפוך רעיון לתוכנה ממשית. למדתי איך לפרק מערכת מורכבת לחלקים פשוטים ויעילים. אני אמנם למדתי על רשתות מחשבים לפני כן, אך הפרויקט הוא למעשה הפעם הראשונה שיצרתי מסגרת תקשורתית בין מערכות שונות ברשת.

בנוסף לידע הגומלי שנרכש לאורך העובדה, שיפרתי את הכלי החשוב ביותר, היכולת ללמוד. מודל הטרנספורמר, והשימוש שלו במודלים שיחתיים בפרט, הוא נושא חדש מאוד. הרבה מהידע נחשב קניין רוחני של חברות ענק, שמנסות ככל האפשר למנוע את ההפצה שלו. לכן הייתי צריך להתמודד עם שיטת למידה חדשה. אין ספר או מאמר שמסביר כיצד כל התהליך עובד. השלבים שרויים בין מאמרים שונים שצריך לאתר. אני סבור שהניסיון עם שיטת למידה זו תתרום לי בעתיד.

סיום הפרויקט אינו מהווה סיום מסע למידת הבינה המלאכותית שלי. אני מתכנן להרחיב את הידע שלי בארכיטקטורת הטרנספורמר. המודל לא רק שימושי למשימות עיבוד שפה טבעית אלא לעיבוד כל סוגי המידע. רשתות מתקדמות שיכולות להפוך משפטים לתמונות מתבססות על טרנספורמרים, שבקום לנבא אסימונים של מילים הם מנבאים אסימונים של פיקסלים. בנוסף, יש רשתות שיכולות לקבל כקלט סוגי מידע שונים כמו טקסט, תמונות והקלטות קול. מודלים אלה מכונים multimodal models, וליצור אחד הוא האתגר הבא שהצבתי לעצמי.

## ביבליוגרפיה

- 4Oh4. (2017, Dec 5). *Where should I place dropout layers in a neural network?* Retrieved from Stack Exchange: <https://stats.stackexchange.com/questions/240305/where-should-i-place-dropout-layers-in-a-neural-network>
- A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*.
- Ajagekar, A. (2021, Dec 16). *Adam*. Retrieved from Cornell University: <https://optimization.cbe.cornell.edu/index.php?title=Adam>
- Alammar, J. (2018, June 27). *The Illustrated Transformer*. Retrieved from <https://jalammar.github.io/illustrated-transformer/>
- Amanatullah. (2023, Sep 1). *Transformer Architecture explained*. Retrieved from Medium: Transformer Architecture explained
- Barnard, J. (2024, Jan 23). *What are word embeddings?* Retrieved from IBM: <https://www.ibm.com/topics/word-embeddings#:~:text=Word%20embeddings%20are%20a%20way,relationships%20among%20the%20corresponding%20words.>
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., & ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*.
- Brownlee, J. (2021, Jan 13). *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. Retrieved from Machine Learning Mastery: <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>
- Cristina, S. (2023, Jan 6). *The Attention Mechanism from Scratch*. Retrieved from Machine Learning Mastery: <https://machinelearningmastery.com/the-attention-mechanism-from-scratch/>

- Cristina, S. (2023, Jan 6). *The Transformer Model*. Retrieved from Machine Learning Mastery: <https://machinelearningmastery.com/the-transformer-model/>
- Deepanshusachdeva. (2023, Jun 14). *Understanding Transformers Step by Step — Word Embeddings*. Retrieved from Medium: <https://deepanshusachdeva5.medium.com/understanding-transformers-step-by-step-word-embeddings-4f4101e7c2f>
- dontloo. (2019, Aug 29). *What exactly are keys, queries, and values in attention mechanisms?* Retrieved from Stack Exchange: <https://stats.stackexchange.com/questions/421935/what-exactly-are-keys-queries-and-values-in-attention-mechanisms>
- Dyouri, A. (2021, Sep 13). *How To Use Templates in a Flask Application*. Retrieved from digitalocean: <https://www.digitalocean.com/community/tutorials/how-to-use-templates-in-a-flask-application>
- Erica. (2021, Jan 2021). *Why do transformers use layer norm instead of batch norm?* Retrieved from Stack Exchange: <https://stats.stackexchange.com/questions/474440/why-do-transformers-use-layer-norm-instead-of-batch-norm>
- Hugging Face. (n.d.). Retrieved from Byte-Pair Encoding tokenization: <https://huggingface.co/learn/nlp-course/en/chapter6/5>
- Jakindah, D. (2023, May 17). *Top P, Temperature and Other Parameters*. Retrieved from Medium: <https://medium.com/@dixnjakindah/top-p-temperature-and-other-parameters-1a53d2f8d7d7>
- Karani, D. (2018, Sep 1). *Introduction to Word Embedding and Word2Vec*. Retrieved from Medium: <https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa>
- Karpathy, A. (2015, May 21). *The Unreasonable Effectiveness of Recurrent Neural Networks*. Retrieved from Andrej Karpathy blog: <https://karpathy.github.io/2015/05/21/rnn-effectiveness/>

- Karpathy, A. (2019, Apr 25). *A Recipe for Training Neural Networks*. Retrieved from Andrej Karpathy blog: <https://karpathy.github.io/2019/04/25/recipe/>
- Karpathy, A. (2022, Sep 12). *Building makemore Part 2: MLP*. Retrieved from YouTube: [https://www.youtube.com/watch?v=TCH\\_1BHY58I](https://www.youtube.com/watch?v=TCH_1BHY58I)
- Karpathy, A. (2022, Sep 7). *The spelled-out intro to language modeling: building makemore*. Retrieved from YouTube: <https://www.youtube.com/watch?v=PaCmpygFfXo>
- Karpathy, A. (2023, Jan 17). *Let's build GPT: from scratch, in code, spelled out*. Retrieved from YouTube: <https://www.youtube.com/watch?v=kCc8FmEb1nY>
- Kazemnejad, A. (2019, Sep 20). *Transformer Architecture: The Positional Encoding*. Retrieved from AmirHossein Kazemnejad's Blog: [https://kazemnejad.com/blog/transformer\\_architecture\\_positional\\_encoding/](https://kazemnejad.com/blog/transformer_architecture_positional_encoding/)
- Khanna, C. (2021, Aug 31). *Byte-Pair Encoding: Subword-based tokenization algorithm*. Retrieved from Medium: <https://towardsdatascience.com/byte-pair-encoding-subword-based-tokenization-algorithm-77828a70bee0>
- Kingma, D. P., & Ba, J. (2014). A method for stochastic optimization. *ArXiv*.
- Mwiti, D. (n.d.). *GPT Instruction Fine-tuning With Keras*. Retrieved from Machine Learning Nuggets: <https://www.machinelearningnuggets.com/gpt-instruction-fine-tuning-with-keras/>
- Norouzi, A. (2023, May 26). *Transformer with TensorFlow*. Retrieved from <https://arminnorouzi.github.io/posts/2023/05/blog-post-13/>
- OPTCausalLM model*. (n.d.). Retrieved from Keras NLP: [https://keras.io/api/keras\\_nlp/models/opt/opt\\_causal\\_lm/](https://keras.io/api/keras_nlp/models/opt/opt_causal_lm/)
- Phillips, H. (2023, May 9). *Layer Normalization*. Retrieved from medium: <https://medium.com/@hunter-j-phillips/layer-normalization-e9ae93eb3c9c>



- Platen, P. v. (2021, Mar 1). *How to generate text: using different decoding methods for language generation with Transformers*. Retrieved from Hugging Face: <https://huggingface.co/blog/how-to-generate>
- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.
- Rush, S. (2022, 1 1). *The Annotated Transformer*. Retrieved from Harvard NLP: <https://nlp.seas.harvard.edu/annotated-transformer/>
- Saeed, M. (2023, Jan 6). *A Gentle Introduction to Positional Encoding in Transformer Models, Part 1*. Retrieved from Machine Learning Mastery: <https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/>
- Sharma, C. (2023, Oct 27). *Powering Flask with Websockets*. Retrieved from Medium: [Powering Flask with Websockets](https://medium.com/@sharmac/powering-flask-with-websockets)
- Singh, V. (2023, Sep 22). *A Guide to Controlling LLM Model Output: Exploring Top-k, Top-p, and Temperature Parameters*. Retrieved from Medium: <https://ivibudh.medium.com/a-guide-to-controlling-llm-model-output-exploring-top-k-top-p-and-temperature-parameters-ed6a31313910>
- Summary of the tokenizers*. (n.d.). Retrieved from Hugging Face: [https://huggingface.co/docs/transformers/en/tokenizer\\_summary](https://huggingface.co/docs/transformers/en/tokenizer_summary)
- Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., . . . Hashimoto, T. B. (n.d.). *Alpaca: A Strong, Replicable Instruction-Following Model*. Retrieved from Stanford University: <https://crfm.stanford.edu/2023/03/13/alpaca.html>
- TecHunter. (2012, Sep 8). *What is the equation used to calculate a linear trendline?* Retrieved from stackexchange: <https://math.stackexchange.com/questions/204020/what-is-the-equation-used-to-calculate-a-linear-trendline>
- Tensorflow. (n.d.). *Keras documentation*. Retrieved from tensorflow: [https://www.tensorflow.org/api\\_docs/python/tf/keras/](https://www.tensorflow.org/api_docs/python/tf/keras/)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*.

Wagner, J. (n.d.). *Chapter 8 Attention and Self-Attention for NLP*. Retrieved from Modern Approaches in Natural Language Processing: [https://slds-lmu.github.io/seminar\\_nlp\\_ss20/attention-and-self-attention-for-nlp.html](https://slds-lmu.github.io/seminar_nlp_ss20/attention-and-self-attention-for-nlp.html)

Weng, L. (2018, Jun 24). *Attention? Attention!* Retrieved from <https://lilianweng.github.io/posts/2018-06-24-attention/>

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., & ... & Zettlemoyer, L. (2022). Opt: Open pre-trained transformer language models. *arXiv*.

## נספחים

### מילון מושגים

עבור מושגים רבים לא נמצא עברות מוסקם, ולעיתים תרגום כלל. טבלה זו נועדה לרכז את משמעותיהם של מושגים ומילים כדי להקל על מי שמסנה לקרוא את הספר.

הגדרה	עברית	אנגלית
יחידה המלכדת מספר שכבות ומאפשרת עיבוד מידע באמצעות מנגנון התשומת לב	טרנספורמר	transformer
מנגנון המאפשר לאיברי הקלט להיות מודעים לאחד השני	תשומת לב	attention
הקלט שהמשתמש שולח למודל שיחתי	הודעה	prompt

הקלט שהמודל השיחתי מעבד (לעיתים קרובות מהווה חלק קטן מההודעה שהשתמש שלח)	טווח הקשר	context window
מילה באוצר המילים של המודל, יכול להיות כמה אותיות, מילה שלמה, סמני פיסוק ומספרים	אסימון	token
אלגוריתם שהופך טקסט רגיל לאסימונים שהמודל יודע לפענח	מקודד	tokenizer
מנגנון שבוחר אסימון מתוך התפלגות ההסתברויות שהמודל מנבא	סמפלר	sampler
תיאור מבנה רשת נוירונים: אילו שכבות קיימות, כמה שכבות ובאיזה סדר	ארכיטקטורת מודל	model architecture
פרמטר ברשת הנוירונים שמתעדכן בתהליך הלמידה כדי לדייק את המודל	פרמטר נלמד	trainable parameter
תהליך יצירת תוכן באמצעות מודל יצרני	הסקה	inference

## דוגמאות לפלטי המודלים

### המודל הראשון

כדי להתחיל כל כתבה המודל יקבל את המשפט הראשון שיכלול את הנושא הכללי שלה וישמש כפתיח. הוא יידרש לכתוב את שאר הכתבה על בסיס המשפט הראשון בלבד. הפשט שנכתב על ידי יודגש בצע, בזמן שהתוכן שמודל יצור לא יהיה מודגש. הגדרות הסמפלר הן  $\text{top\_p}=0.5$   $\text{temp}=1$ .

1. tensions escalate in the middle east as diplomatic efforts fail to ease the conflict. the taliban claimed responsibility for the attacks on the nato-led forces in afghanistan and pakistan on sunday, but said it was a 'massive and dangerous attack'. in the latest scandal that has left the taliban in afghanistan, a u.s.-led

war veteran has claimed responsibility for the attack. scroll down for video . in the latest scandal: afghan security officials have said the taliban's attack has been a 'massive and dangerous attack' in the latest scandal . taliban fighters have claimed responsibility for the attack in the northwest part of the country, which has targeted nato forces in afghanistan and other parts of the country. the taliban claimed responsibility for the attack in the city of kabul. the taliban has claimed responsibility for the attack in the northwest part of the country. the taliban has reportedly been targeted by a 'massive operation' in afghanistan.

2. a natural disaster strikes a coastal town, leaving residents displaced and infrastructure damaged. a government spokesman said the rainfall could have led to an earthquake in the east of the country. the government said the rainfall in northern ireland could have been more expensive than previously thought, but it was only a matter of time before the storm hit. scroll down for video . dangerous: the rainfall is expected to reach at least 150 miles from the northern ireland, leaving residents displaced and infrastructure damaged . a woman said she was treated for severe burns to her body and was taken to hospital where she was taken to hospital where she was treated for serious burns. she was later transferred to a local hospital where she was treated for a shock. the local resident, who asked not to be named, said: 'we were told to stay there. 'we were told to stay away from the area. we had to be very busy to stay there. 'we had to be so busy to the shore.' residents said the rainfall was on the floor.
3. investigations are underway following a major data breach that exposes sensitive information of millions of users. the daily mail revealed how to use a device to track the users images of their phone, including a selfie, a screen that would be used in a specially designed and accessible account. the orange handset would be used in the smartphone app, pictured, to track the users handset by hand . the 'warm' can be used in the device, or any other device, and could also be used in the device. this could also be used in the device, as well as the sensitive and physical scanner for the phone, or other devices. the scanner could be used in the device application process and is expected to be used in the device app. the device is a device that could be used in the device apple device, and could be used in the device apple iphone app. the device is expected to be used in the device apple iphone app .

4. a financial market crash sends shockwaves through global economies, prompting fears of a recession. and a french economy has become the first of its kind in a century. this was revealed in a new report published in the journal nature research, published in the journal nature research, by the us economics association. the report warns that a banking system is likely to cause trouble in europe and is likely to cause trouble in the middle east. the report also warns that the financial crisis is likely to cause trouble in the middle east. 'the new report is not an issue,' said the report, which was published in the journal nature research. the report says that in an economic crisis, the report is likely to cause trouble in the middle east, the rise of a rise in economic growth. the report, published in the journal nature research, said that the rise in financial tensions, which are set to continue in the middle east, could cause trouble in the middle east.
  
5. amidst escalating tensions, world leaders convene for an emergency summit to address the growing threat of cyber warfare. in a debate over ukraine's government's economic plan to shut down the uk government, it was reported that the unemployment rate was the highest since records began in 2007. the plan to shut down the country, which has now been running for more than a decade, was raised by a gdp survey of nearly 200,000 people. it is the highest annual conference in the country, which has now been running for more than a decade. the new plan to shut down the country, which has now been running for more than a decade, will see more than 2,000 people flock to the uk, according to a report from the national academy of sciences. the prime minister said: 'the majority of people are unemployed and people are going to do the same in the next few years. 'these results are going to be unfair and unemployment has been the biggest economic challenge in the country. 'we have had an impact on the uk economy and its people have a strong connection with the global economy. 'we will do everything we can to make sure that the next economic reform is being made in the future. 'we are still in the midst of a debt crisis in the middle of the next few years.'
  
6. in a landmark decision, a global coalition of nations has pledged to take decisive action to combat climate change, signaling a new era of environmental cooperation. the u.s., the unmanned coalition of nations, the nations and the uk, said it would take decisive action to combat climate change. the decision will be

made by the end of the year, and it will be 'must the best thing' for climate change, he said. scroll down for video . scroll down for video . the coalition of nations in nations and the uk has announced plans to impose strict sanctions on climate change, signaling a new era of greenhouse gas emissions . the coalition of nations is planning to take decisive action to combat climate change, but the idea that the u.s. will make up the first of the greenhouse gas emissions is a stronger proportion than any other planet in the world. the coalition will take decisive action to stop climate change in the next decade, the white house said on thursday.

7. in response to mounting pressure, major tech companies have announced sweeping reforms to address concerns over data privacy and security. the changes to a new network app, called internet connectivity, are also aimed at sharing information about what it has done. the system is designed to measure the number of personal information about a person's connectivity, and apps that allow people to view personal information about their personal information and personal information online. the new network app is designed to measure the number of information about a person's connectivity and personal information on the app, and apps that allow people to view online . the system is designed to measure the number of data networks, including what people say is about data privacy and privacy, and where people are privacy. apps that allow people to view online and send messages about privacy and privacy are being allowed to view online. a new app, called internet connectivity, has already announced plans to launch a new app online on the app, and it is designed to measure the number of data and privacy issues. it is designed to measure the number of phone calls and the phone is expected to launch at the end of 2015.
8. as tensions flare between rival factions, a fragile ceasefire in a war-torn region hangs in the balance, raising concerns about the prospects for lasting peace. the president, who wants to be a leader of the city, is hoping to continue talks with other parties to stop and towards israel, which will begin on the other side of the river thames in the midst of a decade. the talks are aimed at the united nations security council and president barack obama. scroll down for video . peace talks between rival factions and security council members have been held by u.s. forces in the balance between rival factions . the official, who spoke to the media forum on sunday, said he would consider . the referendum to be a

leader of the islamic state group . the palestinian militants have been involved in the battle of the . city of baghdad. the statement said that the situation in the city was 'totally significant' and that the army was prepared to carry out an 'exposure of peace' against the islamic state, the statement said. the talks will begin on monday in the midst of a war with israel and the u.s. in the middle east. the conflict is expected to last until november, but the group is believed to have pulled out of the stricken government, with the u.s. expressing concern about the militants' response. 'the government is going to know what the syrian people are doing on the basis of peace and security,' the statement said. 'we will take all necessary steps to make sure that they are ready to accept the government and take the responsibility of the un security council.' the syrian government is preparing to 'make a peaceful resolution to the war, and this is not the first time the conflict has occurred,' the statement said. the announcement comes as a u.n. summit in syria was held on friday. the u.s. has taken up arms with the un urgent need to step down from the u.n. general assembly to end the civil war.

9. following a string of protests, lawmakers are considering sweeping reforms to address systemic issues of inequality and injustice. and a major concern for a local politician who has spoken of his comments in a bid to become a political force in a bid to promote the end of the year. in an interview with mailonline, a major concern for a campaign group, mr cameron said that the proposal was unfair. 'people have to be honest about what they want to do,' he said. scroll down for video . 'worldrunner': protesters have been considering sweeping reforms to end the violence against civil rights campaigners . the prime minister said the measures were 'prepared' to be 'humiliated' and said he would not accept the changes . mr cameron told mailonline: 'this is a disastrous decision. the proposal is unfair. the proposal will take place in a bid to boost public spending and increase the scale of the violence.' the bill would be reduced to 20 per cent of the vote and would mean it would be reduced to 24 per cent of the vote. the move comes after it emerged that angry voters who want to reject the idea of a vote are expected to vote for a vote on the issue. scroll down for video . david cameron said the proposal was 'prepared' to support a vote on the issue of gun violence . the scottish government has urged the government to shut down the bill. but the measures could be cut to 20 per cent of the vote on the issue. mr cameron said he would vote on the proposal of a vote on the issue. the prime minister said he would be 'delighted' to vote on the issue of gun violence and

increase the ukip leadership's plan to make the proposal 'prepared to happen'. but he said the plan would 'delighted' the measure to curb the future of the ukip leadership. he said: 'there is no need to be done to put the issue in front of the ukip debate on the issue of gun violence. 'there is no need to have to be done to stop the action from happening again.'

10. with elections looming, candidates are intensifying their campaigns, vying for voter support in key battleground states. the country is set to open its next election to a vote in the next election. the campaign has been widely seen as a party that holds a key vote in the democratic elections. the polls will show that 24 per cent of voters vote on the issue of candidates. voters are intensifying their campaigns, and the party is set to open its next election to a vote in key battleground states. scroll down for video . party voters are intensifying their votes on the issue of elections in key battleground states, which holds a key vote in key battleground states . the polls will show that just 54 per cent of voters vote on the issue, according to the polls . voters are expected to make their vote a key vote in key battleground states, where 24 per cent vote on the issue of candidates. polls show that 42 per cent of voters vote on the issue, compared with 42 per cent who vote on the issue of elections. polls show that 42 per cent of voters vote on the issue. in . key battleground states, the vote will likely be held on thursday by the . polls today, and the polls show that voters vote on the issue of elections . 'this is an issue that has been held on the issue of elections,' the polls show. polls show that 42 per cent of voters vote on the issue of elections on the issue of elections, while 32 per cent said the issue of elections would not be held on thursday by the end of the week. 'what i say is the party is in the best interests of elections,' the polls showed. 'it's a crucial question. it is the party that will be won by the end of the year. 'we are going to do something about elections and that is what we are going to do. 'the polls are going to be held on the same standards we are going to do. but i think we have to do that. 'we are going to do something about elections. elections are going to be held on the same standards we are going to do. 'they are going to be held on the same standards we are going to do.'

11. with the rise of social media, misinformation continues to spread unchecked, raising questions about the role of platforms in shaping public discourse. the scandal comes after a report by a politician revealed that he had been



unchecked at a conference on the issue of immigration reform in 2012. the report, published by the washington post, found that the issue of immigration reform is likely to cause unchecked debate over immigration reform. but it is not known if immigration reform is a crime and the deportation of the immigration reform workers would have a power to spread unchecked terms. scroll down for video . a report by the national education association (nbc) revealed that immigration reform is a major issue for immigration reform, with one in three of the top 10 children in the country, the report said. 'there is no evidence that the u.s. is on the issue of immigration reform,' the report said. the report also found that the report was a serious issue for immigration reform. 'the report is a serious issue for immigration reform,' the report said. 'the report was published by the national education association of children and families that were forced to resign.' the report, published on friday, revealed that the report's authors, including the head of the national education association (nbc) said that the report was 'disappointed' and 'concerned that this report is the real reason why the report was 'unchecked'. the report says that a report by the national education association of children and families has been 'willing to provide a clear and positive response to the report'. the report was released in june, with the report saying that 'the report is a serious issue for the world, the report will be presented in a more conservative case for the national education association. the report will also reveal that the report will be used to interview immigration officials in a press conference to try to interview the report's authors. it is not known whether the report will be used in the report's release. the report also reveals that the report will be used to report a crime on the website.

12. in a bid to revitalize the economy, governments are rolling out stimulus packages and infrastructure projects to spur growth and create jobs. the bidding system has been set up to invest in the economy and spend millions of pounds of benefits. the company has been accused of building up a growing economy of selling oil in a country that has fallen to its value since it was launched in 1998. the bidding system has been set up to help fund global growth and spend growth in the country, the report said. it is unclear how many more taxes are being sold on to foreign banks . the bidding system has been set up to invest in the economy and spend millions of pounds of taxpayers money on expensive goods. the money is being spent on the project, and the £1.5billion banking system is

set to rise to £1.5billion in the next three years. the scheme will see a growth of 37.5 per cent in the coming year and the total share of the value of taxpayers money is up by £1.5billion. the bank of england will announce that it will cut to more than £1billion in 2015. it will also boost the project on its £1.5billion package of the government. a bid to fund the recovery will be set to rise to £1.5billion. the figures are expected to be made by chancellor george osborne, who has warned the move will be more serious than a £1.5billion spent on foreign banks.

13. as vaccination rates stagnate, health officials launch campaigns to combat vaccine hesitancy. the average person with vaccination is just two days away from home, but their new outbreak is now being aimed at the united nations and other countries that have warned that the virus is becoming increasingly infected. the vaccine was used to help the vaccine improve the vaccine in response to the virus. it is not clear how the vaccine works on vaccine, but the vaccine is already being built to combat vaccine hesitancy. vaccination rates are the most likely outbreak of vaccine hesitancy - an effective vaccine that helps infect vaccines and fluids - in a way that affects vaccination . vaccine techniques are also the most common vaccine that affects vaccination. vaccine in vaccine science has become a viral threat, with vaccine use in vaccination rates among vaccination centres and a vaccine that has also been developed in the united states. vaccination rates are usually up from 60 per cent in vaccination to 60 per cent in vaccination rates. vaccination rates are also higher in vaccination rates and vaccines, such as the vaccine, which is a vaccine that needs to be delivered to vaccinated vaccination centres.
14. amidst rising inflation, central banks announce interest rate hikes. this will mean a £200million fall in debt by £12billion a year. this is expected to rise by the end of the year. last night, a bank of england computer systems announced the highest rate of income in europe and the u.s. bank of england. the proposal would see the financial fair share rise as the bank of england has not done enough. this would be a boost for banks across the country, with the economy set to rise by the end of the year. scroll down for video . in the last week, the bank of england will start investing in a £200million increase in debt by £12billion a year . this will be a boost for banks in the bank of england, where the bank of england will move into a £200million cash stock. the financial fair share is

expected to rise by the end of the year. it will see the bank of england pay its fair share of debt as it will increase by £12billion a year in debt by £112billion a year. the bank of england will spend on financial support and banks across the country on the debt of the year, and is expected to increase by £12billion a year to £12billion a year. but banks have made significant cuts to the bank of england and bank of england, and are expected to see a £100billion a year increase in debt as it continues to grow. bank of england governor jimmy carter, pictured, will see a boost in debt as a result of a growing economy over the next two years . bank of england governor jimmy carter will see a boost in debt as the bank of england will turn to a total £12billion a year . financial support, which is expected to rise by £12billion a year . bank of england governor jimmy carter will see a boost in debt . as banks and banks will see a boost in debt .

15. following a diplomatic breakthrough, historic peace talks commence in the region. the leader of the islamic state in syria will be drawn up to join the terror group in syria and iraq, the islamic state of iraq and syria and syria will be brought to the uk. the white house has said that the united nations has been struggling to deal with the islamic state of iraq and syria and the islamic state of iraq and syria for the first time since its uprising in the arab world began. this is a country that is going to fight in syria and iraq and syria. it is the latest in a series of political and political crisis in syria and iraq. the new leader of the islamic state is now hoping to join the islamic state of iraq and syria for the first time since it was introduced in 2001. scroll down for video . this week the white house has announced that the new leader of the islamic state of iraq and syria will be brought to the uk for the first time since it was introduced in 2001 . the new leader of the islamic state of iraq and syria will be brought to the uk for the first time since the fall of the islamic state of iraq and syria in 2001. the new leader of the islamic state of iraq and syria will also have to wait for the new leader of the islamic state of iraq and syria to be deployed to syria and syria to help with the international community. in the new leader of the islamic state, the leader of the islamic state of iraq and syria will be given to the national security council, as well as the u.s., as part of the new leadership of the islamic state of iraq and syria. in the new leadership, the leader of the islamic state of iraq and syria will return to the uk for the first time since the fall of the islamic state of iraq and syria.

## המודל השני

כל התשובות נוצרו עם סמפלר בעל  $\text{temperature}=0.5$  ו  $\text{top\_p}=0.5$ .

1. Q) What is the main cause of climate change?
  - A) The main cause of climate change is the burning of fossil fuels, such as coal, oil, and natural gas. This burning of fossil fuels releases carbon dioxide into the atmosphere, which traps heat in the atmosphere and leads to a rise in global temperatures.
2. Q) What is the value of 1+1?
  - A) The value of 1+1 is 1.
3. Q) Give me a list of five tips to help combat climate change.
  - A)
    1. Reduce your energy consumption by using energy efficient appliances and turning off lights when not in use.
    2. Reduce your water consumption by using low-flow fixtures and shower heads.
    3. Install solar panels to capture the sun's energy and convert it into electricity.
    4. Utilize natural resources to generate electricity, such as rainwater harvesting and wind turbines.
    5. Plant more trees to absorb carbon dioxide from the atmosphere.
4. Q) Write a short story about a man walking his dog.
  - A) John walked his dog, a curious and adventurous one. Every day was the same: running around the yard, enjoying the sunshine, and playing in the park. But today was something different. He had just gotten out of the house and was walking his dog, a curious one that could do more than just show him around.

One day, John decided to take his dog for a walk. As soon as he got there, he noticed something strange in the distance. It was a beautiful puppy, with a mysterious, magical tail that could fly.

John took the puppy home and the puppy was safe. But soon, the puppy had vanished.

John was devastated. He searched the yard for it, but couldn't find it anywhere. After a few days, he finally found it. He was sure he'd found it there somewhere else.

John had found it.

5. Q) What is the capital of Israel?

A) The capital of Israel is Jerusalem.

6. Q) Explain the dangers of climate change.

A) Climate change is having a significant impact on the environment and health of individuals and communities. It is causing an increase in global temperatures, leading to an increase in global temperatures, an increase in the intensity of natural disasters, and an increase in the risk of diseases and other health issues. Additionally, it is causing sea levels to rise, extreme weather events to become more frequent, and an increase in the risk of species extinction.

7. Q) Give an example of a fruit that grows on trees.

A) An example of a fruit that grows on trees is a banana.

8. Q) Why is switching to a sustainable lifestyle important?

A) Switching to a sustainable lifestyle is important because it can help to reduce your carbon footprint, improve your health, and reduce your risk of developing chronic diseases. Additionally, switching to a sustainable

diet can help to reduce your risk of developing chronic diseases such as diabetes, heart disease, and certain cancers.

9. Q) What are the benefits of using the Python programming language?

A) The benefits of using Python programming languages include improved code readability, improved code readability, improved code readability, improved code readability, and improved code readability. Python also provides a more secure and efficient code readability, as well as improved code readability for better readability.

10. Q) What is the longest river in the world?

A) The longest river in the world is the Nile River, which stretches for over 4,000 miles and covers an area of over 2,000 square miles.

11. Q) What are the primary factors leading to global warming?

A) The primary factors leading to global warming include the burning of fossil fuels, deforestation, agricultural activities, industrial activities, and human activities.

12. Q) Name a vegetable commonly found in salads.

A) A popular vegetable commonly found in salads is lettuce.

13. Q) Discuss the benefits of renewable energy sources over fossil fuels.

A) Renewable energy sources provide a number of benefits over fossil fuels. These include the ability to generate electricity without the use of fossil fuels, as well as the potential to reduce greenhouse gas emissions and combat climate change. Additionally, renewable energy sources are a clean, sustainable source of energy, which means they can be used to power plants and other forms of renewable energy sources.

Furthermore, renewable energy sources are often cheaper than fossil fuels and can be used to power more complex projects and industries. Finally, renewable energy sources are often more secure and require less maintenance than fossil fuels, making them a valuable investment in the long term.

14. Q) Explain the concept of supply and demand in economics.

A) Supply and demand in economics is the concept that the total demand for a product or service is determined by the supply and demand that is being generated from the product or service. This means that the higher demand for the product or service increases the supply, while the lower demand for the product or service decreases. This concept is important because it helps to explain the relationship between the supply and demand in economics and how to better understand the dynamics of the market.

15. Q) What are the major causes of air pollution in urban areas?

A) The major causes of air pollution in urban areas include the burning of fossil fuels, the release of pollutants in the atmosphere by industrial activities, the use of pesticides and herbicides in the urban environment, the urban environment's poor air quality, and the lack of green spaces.

16. Q) Name three major Renaissance artists.

A) The three major Renaissance artists are Leonardo da Vinci, Vincent Van Gogh, and Rembrandt.

17. Q) Who wrote "To Kill a Mockingbird"?

A) The Harper Lee novel "To Kill a Mockingbird" was written by Harper Lee.

18. Q) Who discovered penicillin?

A) The first person discovered penicillin in the late 1940s was Albert Einstein.

19. Q) Who is the current Secretary-General of the United Nations?

A) The current Secretary-General of the United Nations is currently in charge.

20. Q) Write a happy birthday letter to a friend.

A) Dear friend,

Happy birthday to you on this special day. May this year bring you a day of joy and laughter that will stay with you forever.

Happy birthday to you,  
[Your Name]