איתור שגיאות במסמך שעבר OCR

ניצן בר – 302221338
רוי לוי – 313577611

- ניתן למצוא את הפרויקט ב$Github$ בקישור הבא:

רקע ומטרה

מטרתו של הפרויקט הינו לזהות שגיאות שנוצרו עקב דיגיטיזציה של מסמכים באמצעות OCR. בידינו, כחלק מן המידע המסופק לצורך הפרויקט, מסמכים שעברו OCR ואת המקור שלהם בפורמט PDF.

לצערנו, סריקת OCR אינה נאמנה למקור וקיימים מקרים בהם ישנן שגיאות עקב הפעלת תהליך זה על המסמכים. מטרתנו בפרויקט היא לאתר ולהתריע על השגיאות הללו.

להלן דוגמאות נפוצות שנתקלו בהן בעת ביצוע המשימה:

1. שגיאות כתיב
   1.1 החלפה בין נ' לב' : מדינת ← מדיבת.
   1.2 החלפה בין ח' לה': חוק ← הוק.
   1.3 החלפה בין ין ל-ץ: מונופולין ← מונופוליץ.
2. בלבול בסימני פיסוק
   2.1 פסיק (,) ← גרש עליון (').
   2.2 נקודה (.) ← לפסיק (,).
3. שגיאות בהערות שוליים
   3.1 התשכ"ג-1963' ← התשכ"ג-11963

שיטת העבודה בה בחרנו

אנו בחרנו להתמקד בשיטות למידת מכונה ולמידה עמוקה על מנת לפתור את הבעיה.
נפרט על השלבים העיקריים בפתרון המשימה (נפרט בהרחבה בהמשך):

- סריקת קורפוס קבצי הXML מהקבצים שניתנו לנו, כלומר ביצוע חילוץ של הטקסט מהקבצים הללו.
- ביצוע פעולת $scraping$ מאתר ויקיטקסט על מנת להביא עוד חוקים ולהעשיר את המידע שלנו.
- ביצוע עיבוד מקדים הכולל שימוש בביטויים רגולריים על מנת לנקות את הטקסט ולהעביר אותו לפורמט אחיד.
- מעבר באמצעות קריאה צמודה על כ300 מסמכי $DOCX$ והכנת מילון של טעויות נפוצות ומספר הפעמים שהופיעו.
- ביצוע פגימה מכוונת בטקסט באמצעות המילונים שהכנו וכן תיוג המילים שנפגמו בהתאמה.
- ביצוע טוקניזציה והכנת המידע לצורה מתאימה כך שיהיה ניתן להכניס אותו למודלים.
- הזנת המידע ולמודלים של למידת מכונה ולמידה עמוקה.
- שימוש במודל בעל התוצאות הטובות ביותר (מודל למידה עמוקה) על מנת לחזות את הטעויות בקבצי ה$DOCX$.
- תיוג המילים החשודות לפי התוצאות שהמודל חזה.

תיאור הפרויקט במונחים של מדעי הרוח הדיגיטליים

על מנת ללמד את המודל שלנו נעזרנו ב**מסמכים חצי מובנים**, כלומר שימוש קבצי הXML וכן חילוץ מהם את **תוכן** התגיות הרלוונטית (חילוץ הטקסט מהתוויות). בנוסף נעזרנו ב**מסמכים לא מובנים** (קבצי הDOCX), כלומר תוצרי מסמכי הOCR על מנת להכין **מילונים** של שגיאות נפוצות.

נעזרנו **במקור חיצוני** (ויקיטקטס) כדי להרחיב את המידע שברשותנו על ידי שליפת חוקים נוספים מאתר זה. בנוסף נעזרנו ב**ביטויים רגולריים** על מנת לנקות את הטקסט ולהכין אותו לשלב ה**טוקניזציה**. על מנת לבדוק שהמידע שברשותנו תקין לאחר העיבוד המקדים נמצא תקין ביצענו **Data Exploration** מסוגים שונים כגון חקר **ngrams** ו**wordClouds**.

ביצענו **קריאה צמודה** על כ300 מסמכים שעברו OCR על מנת להכין **מילונים** של שגיאות נפוצות.

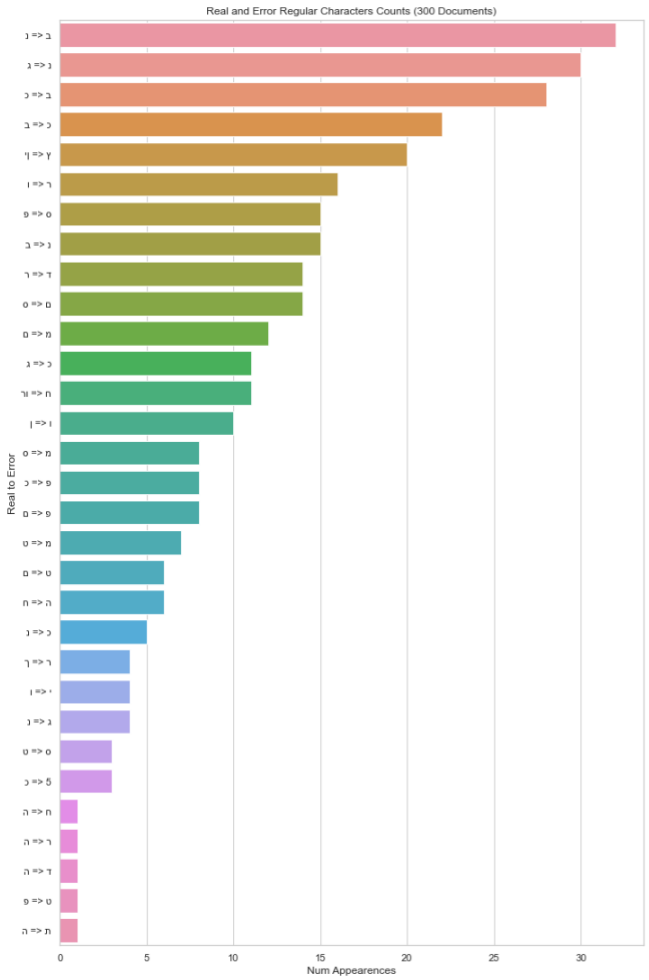
נעזרנו ברשת נוירונים שמטרתה ללמוד את ה**הקשרים** בין המילים וכן מנסה לתת משקל דומה למילים אשר קרובות ב**משמעות הסמנטית** או באות בד"כ אחת אחרי השנייה (**ngrams**). נעזרנו ב**תיוג בינארי** על מנת לתייג את המילים החשודות לשגיאות.

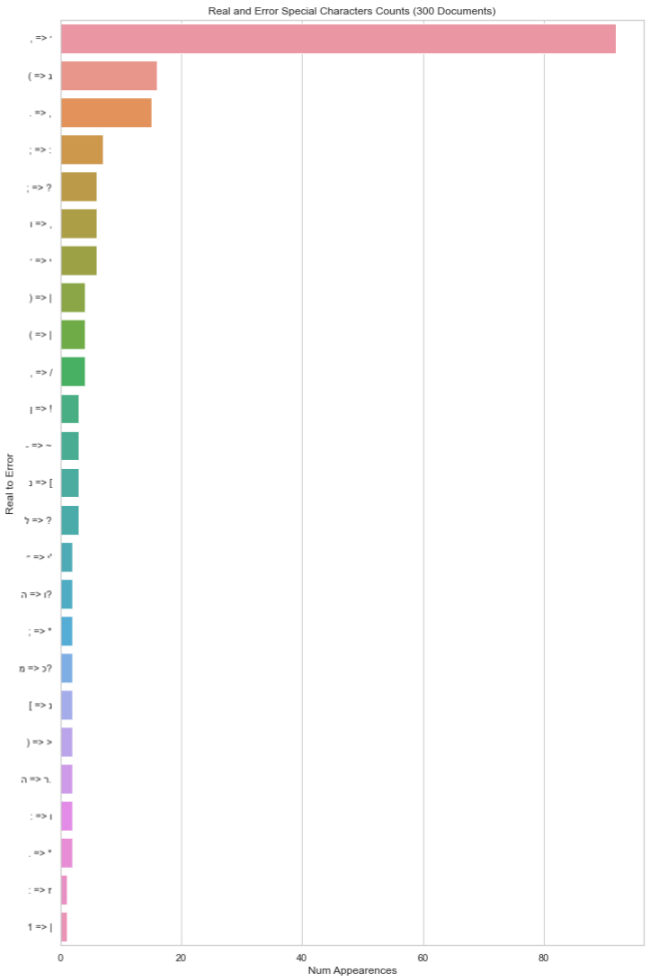
פתרון הבעיה

כעת נסביר על השלבים של תהליך העבודה שלנו וכיצד הגענו לתוצאות שאותן נפרט בהמשך.
נציין תהליך העבודה היה ארוך והדוח הנ"ל אינו מסוגל להכיל את כולו, לשם כך התהליך השלם מפורט במחברת הג'ופיטר הנמצאת בתיקיית הפרוייקט. נציין כי בנוסף לכלל השלבים הנמצאים במחברת, לקחנו את השלבים הרלוונטיים על מנת להכין סקירפט פשוט שעושה שימוש במודל שנוצר במחברת.

שלבים מקדימים

- המרת קבצי doc לקבצי $docx$ לשם נוחות שימוש בספריות רלוונטיות בתוכנית (כלל הקבצים המומרים מסופקים עם הפרוייקט).
- קריאה ומיפוי לפי מזהה חוק של קבצי הXML, הוצאת הטקסט מהקבצים הנ"ל.
- שאיבת מידע נוסף, כ1993 חוקים נוספים מאתר ויקיטקטס.
- ביצוע עיבוד מקדים רלוונטי על הטקסט. לדוג' הוספת רווחים בין מירכאות במקומות שצריך, החלפת מירכאות מיוחדות במירכאות בעברית, חילוץ סימנים בין סוגריים מרובעות ועוד...
- מעבר על כ300 מסמכים באמצעות קריאה צמודה על מנת להכין מילונים של שגיאות נפוצות וכמות הופעתן. ניתן לראות דוג' להיסטוגרמה בגרף הבא:





- פגימה מכוונת בטקסט החוקים באמצעות המילונים שהכנו (פירוט על פונקציית הפגימה בהמשך).
- חלוקת המידע ל3 סטים: אימון, מבחן ולידציה, כמו כן התאמתו לצורה אחידה לצורך הכנסתו למודלים (יפורט בהמשך).
- יצירת טוקניזר הממפה מילים למספרים. הכנסנו את כלל המילים שנמצאות בסט האימון וכן מילים נוספות שנראו לנו רלוונטיות כגון חודשי וימי השנה, שנים עבריות החל מ1938 ועוד...
- התייחסות לבעיה של חיזוי השגיאות כmultilabel classification וכן הזנת המידע למודלים מוכנים של למידת מכונה.
- הכנת מודל למידה עמוקה, שימוש במטריקות מותאמות אישית על מנת לבחון את ההתקדמות בזמן האימון, הכנת פונקציית הפסד מותאמת אישית על מנת לבצע אופטימיזציה למודל.
- בחינת התוצאות, בחירת המודל הטוב ביותר ושמירת המשקולות שלו.

פירוט על פונקציית הפגימה

פונקציית הפגימה עשתה שימוש במילונים שהכנו מקודם המכילים טעויות נפוצות ואת כמות הפעמים שהופיעו. על מנת ליצור פגימה בטקסט בצורה רנדומלית "חכמה" נעזרנו בהתפלגות גאוס. בנינו התפלגות גאוס על פי כמות המופעים של כל טעות וכל פעם דגמנו אחת. התוצאה היא שתחילה נדגום טעויות יותר נפוצות ולאט לאט נתקדם לפחות. כל הבטחנו רנדומליות אך עם זאת נאמנות לסוג מידע שאנו מעוניינים לחזות עליו. בנוסף קבענו שאנו נפגום ב20% מכל מסמך כיוון שלאחר בחינת 300 המסמכים זה היה נראה יחס המילים הפגומות לעומת התקינות.

עבור כל מילה שפגמנו, סימנו במטריצה ייעודית 1 במקום בו המילה הופיעה, יתר המילים שאינן נפגמו סומנו כ0. למעשה, על המודל שלנו לחזות עבור כל מסמך את המטריצה הנ"ל עבור כל מסמך.

<u>התאמת המידע לצורה אחידה</u>

כיוון שאנו נעזרים במודלים על מנת לחזות את השגיאות אנו חייבים להמיר את המידע שיכיל צורה אחידה (בעיקר תקף ללמידה עמוקה). לשם כך קבענו ערכים קבועים למספר המילים בכל משפט וכמות משפטים מקסימלית לכל מסמך. הערכים שקבענו הם 50 ו200, כלומר כל משפט בסט האימון יכול להכיל 50 מילים (מופרדות ברווחים) ויכולים להיות בו 200 משפטים (את היתר קטענו או ריפדנו באפסים במידת הצורך). הערכים הללו נקבעו עקב חקר על הכמויות המקסימליות והממוצעות של הערכים וכן ניסיון מעשי מבחינת היבטים של זמן חישוב, זיכרון ויעילות התוצאות.

בחלק של החיזוי על מסמכי הdocx, כיוון שאנו רוצים לחזות על כמה שיותר מידע, נאלצנו לשמור מיפוי לכל מסמך לכמה חלקים הוא יכול להתחלק עקב הערכים שקבענו ולמפות את המסמך חזרה בעת התיוג בתגית השגיאה. לדוג' נניח שמסמך הכיל 300 משפטים אז הוא יתמפה ל2 חלקים (batches).

<u>שלבי התוכנית העיקרית</u>

השלבים הללו מצויים גם במחברת וגם בקובץ הפייתון, מטרת השלבים הללו היא להשתמש במודל שאומן לצורך חיזוי השגיאות.

- קבלת נתיב לתיקייה המכילה קבצי docx וקריאתם.
- ביצוע עיבוד מקדים (כפי שבוצע קודם) על הטקסט של קבעי הdocx.
- המרת המידע לגדלים קבועים וביצוע טוקניזציה וכן שמירת מטריצה אשר תאפשר מיפוי חזרה לגדלים המקוריים של הקבצים.
- הזנת המידע למודל שאומן מראש וחיזוי שחשודות שמכילות שגיאות.
- סימון המילים החשודות לפי חיזוי המודל, שימוש במטריצת המיפוי לצורך סימון כלל הקבצים בצורה נכונה.
- שמירת הקבצים המסומנים בפורמט txt בתיקייה ייעודית.

<u>הערכת התוצאות ומדדים</u>

הדרך שבה מדדנו את טיב התוצאות באה לידי ביטוי בשני אופנים:

1. בחינה אוטומטית על סט האימון/ולידציה/מבחן.
2. קריאה צמודה של כ10 מסמכים ובחינת התוצאות.

נדבר תחילה על השיטה הראשונה. כיוון שהיה לנו מידע מתויג היה באפשרותנו להעריך את התוצאות של המודל על מספר רב של מסמכים בצורה אוטומטית. נציין כי אנו נדבר על מודל הלמידה עמוקה, כיוון שהניב את התוצאות הטובות ביותר.

דבר חשוב שיש לשים לב אליו זה שהמספר המחלקות שיש לנו הוא 2 (שגיאה/לא שגיאה) וכן מספר השגיאות נמוך באופן משמעותי ממספר המילים התקינות. הדבר נעשה מרצון לשקף את המציאות במסמכי ה*docx*. בגלל הבחנה זו, עלינו להסתכל על מדדים נוספים מלבד *accuracy* אשר בד"כ בשימוש. זאת מכיוון שיכול להיות דיוק גבוה עקב חיזוי רק של מילים תקינות (מתויגות ב0).

על מנת לבחון את המודל השתמשנו במדדים נוספים שנקראים *precision* , *recall*. שני המדדים הללו נותנים לנו מידע יותר מוחשי על האיכות של המודל שלנו. נציין כי על מנת לבחון שהמודל שלנו אכן לומד כתבנו פונקציות *callbacks* מיוחדות אשר מחשבות את המדדים הנ"ל לאחר כל איטרציית אימון.

בטבלה הבאה מובאות התוצאות עבור סט האימון וולידציה לפי המדדים שציינו לעיל:

	Accuracy	Precision	Recall
Train Set	0.9966	0.9227	0.8378
Validation Set	0.9965	0.9145	0.8251

ניתן לראות כי קיבלנו תוצאות גבוהות מאוד עבור סט האימון והולידציה וזה אכן אפשר לנו לדעת כי המודל שלנו מתנהג בצורה הרצויה.

כעת נדבר על התוצאות עבור מסמכי ה*docx*. לקחנו כ10 מסמכים, ביצענו את החיזוי ולאחר קריאה צמודה קיבלנו את התוצאות הבאות:

	True	False
Positive	75	8
Negative	---	13

נסביר מה כל תא בטבלה מסמל:

- *True Positive* – מספר השגיאות שהמודל זיהה שאכן שגיאות.
- *True Negative* – מספר המילים התקינות שהמודל זיהה שאכן תקינות.
- *False Positive* - מספר המילים שהמודל זיהה כשגיאות אך הן תקינות בפועל.
- *False Negative* – מספר המילים שהמודל זיהה כתקינות אך היו שגיאות בפועל.

התא שנשאר ריק מציין את המילים שאכן תקינות וכן המודל זיהה זאת (ישנן מילים רבות כאלו אז לא ספרנו אותן).

<u>דוגמאות חיזוי:</u>

מקור	תוצר	האם השגיאה אותרה?
ביום י"ב בטבת...	ביום י"ב בטבח...	המילה "טבח" זוהתה ע"י המודל כשגיאה בצדק.
פגיעה בכבוד דגל המדינה...	פגיעה בכבוד דגל המדינה...	המילה "דגל" זוהתה כשגיאה למרות שהינה מילה תקינה.
שר הפנים הממונה...	שר הפגים הממונה...	המילה "הפגים" זוהתה ע"י המודל כשגיאה בצדק.

<u>הנחיות שימוש בכלי</u>

כפי שציינו, מעבר למחברת ג'ופיטר, הוספנו סקירפט שבאמצעותו ניתן להריץ את המודל על קבצי *docx* ולקבל חיזוי היכן מצויים בהם שגיאות. אנו נסביר כיצד להשתמש בכלי הנ"ל (להסבר נוסף ניתן להסתכל בקישור ה*Github* שצורף לעיל).

1. וודאו כי יש ברשותכם פייתון 3.6 ומעלה (אנו ממליצים להשתמש ב*anaconda*).
2. הורידו את ספריית הקוד מהקישור <u>הבא</u> (ע"י *git clone* / הורדת קובץ *zip*).
3. הריצו את הפקודה הבא לצורך התקנת הספריות הרלוונטיות:
```
pip install -r requirements_script.txt
```
4. פתחו טרמינל בספריית הפרויקט וצרו תיקייה עם קבצי docx בהם תרצו למצוא שגיאות.

5. הריצו את הפקודה הבאה בטרמינל כאשר אתם מספקים נתיב לתיקייה המכילה את קבצי הdocx:

```
python ocr_predict_errors.py [path_of_your_docx_folder]
```

אם ברצונכם לתת שם לתיקייה בה יופיעו קבצי הטקסט, הריצו את הפקודה הבאה יחד עם השם:
```
python ocr_predict_errors.py [path_of_your_docx_folder] -sn [your_custom_name]
```

הפלט יופיע תחת תיקיית בת של תיקייה בשם text_files יחד עם חותמת הזמן והשם שסיפקתם. כל קובץ בתיקייה הנ"ל הינו בפורמט txt ומכיל את הטקסט של קבצי הdocx כאשר כל מילה אשר חשודה שמכילה שגיאה מסומנת בתגית הבאה: <e>.

סיכום, מסקנות והצעות לשיפור

ניתן לראות כי את רוב השגיאות המודל זיהה בצורה נכונה, כ85% מהשגיאות שהמודל זיהה הן אכן שגיאות וכ92% מכלל השגיאות זוהו. לפי אבחנה שלנו, המודל מזהה לרוב מילים אשר אינו מכיר כשגיאה, כלומר מילים אשר אינן קיימות בטוקניזר. שמנו לב כי אנחנו מצליחים לתפוס הרבה מהשגיאות הנפוצות גם בהערות השוליים וגם בתוכן הראשי, זאת מאחר שהוספת המידע מויקיטקסט עזר לנו להגדיל את המילון שלנו וגם עזר למודל ללמוד מילים ורצפים אשר מופיעים בעיקר בהערות שוליים.

ישנן המון דרכים שאנו חושבים שבאמצעותן ניתן לשפר את המודל אך לא מימשנו עקב קוצר זמן. נמנה חלק מהן כאן:

- הוספת עוד מידע חיצוני – ניתן להוסיף עוד חוקים לדוג' או אפילו טקסטים פשוטים בעברית, על מנת שהמודל ילמד יותר את ההקשרים בין המילים.
- שימוש במודל שאומן מראש – לפעמים לא צריך להמציא את הגלגל מחדש, יש מודלים חזקים שכבר אומנו בעבר וניתן לעשות בהם שימוש (לדוג' *BERT*) ולהתאים אותם לבעיה שלנו.
- ביצוע עיבוד מקדים יותר מקיף – ניתן למצוא בטקסטים עוד אנומליות וחלקים שניתן להוסיף לעיבוד המקדים לצורך חיזוי טוב יותר.
- מעבר על עוד מסמכים ומציאת שגיאות נפוצות – כפי שאמרנו, ביצענו קריאה צמודה של 300 מסמכים ומידלנו את השגיאות הנפוצות. התהליך הנ"ל לקח זמן רב ואפילו לא עברנו על 10% מכלל המסמכים. אנו סבורים כי ניתן להגיע לתוצאות טובות יותר כאשר יהיה לנו מילון גדול יותר של שגיאות נפוצות.
- שימוש ב*OCR* נוסף ומילון חיצוני – ניתן להוסיף עוד "שכבה" לפני התיוג של מילה כפגומה ולשלב מקור נוסף שיוודא האם מילה היא פגומה. לדוגמא, ניתן להכין מילונים שנוצרו עקב סריקה של מודל *OCR* נוסף ולחזות לפי המילונים הללו האם מילה היא פגומה. כמו כן ניתן למשקל את התוצאות השלב הנ"ל עם החיזוי של המודל ולפיכן להעריך האם מילה היא פגומה או לא.

לסיכום, למרות שהעבודה הייתה מאתגרת היא בהחלט הייתה מלמדת וחשובה מבחינתו. למרות שתמיד יש מקום לשיפור אנו סבורים שעמדנו במשימה בצורה טובה וסיפקנו תשתית איכותית שבאמצעותה ניתן למנף את הפרויקט הלאה.

בחרנו לקחת את הפרויקט הנ"ל מכיוון שהנושא ריתק אותנו ובמיוחד השימוש בלמידה עמוקה לצורך חזיית השגיאות. אנו מרגישים שהקשיים השונים שליוו את העבודה רק חיזקו את הידע שלנו. למדנו את החשיבות של עיבוד מקדים, בחינת המידע וביצוע ניתוחים מעמיקים, כיצד לעבוד עם מודלים של למידת מכונה ולמידה עמוקה וכיצד להתאים אותם לצרכים שלנו. בנוסף האתגר של לנסות רעיונות שונים על מנת לשפר את המודל הצריך מאתנו חשיבה יצירתית ונהנינו לחוות מקרוב את השימוש של *OCR* בהקשר של מדעי הרוח הדיגיטליים.

שמחנו לקחת חלק בקורס ולהכיר את התחומים השונים במדעי הרוח הדיגיטליים ואנחנו בהחלט מרוצים שיצא לנו לעשות עבודה שהבענו בה עניין רב.