המכללה האקדמית ספיר

ספר פרויקט

OCR ERROR DETECTOR

בשיתוף פעולה עם

מגישים

בדיע אבו פריח ת.ז. 312217854

מלאק אבו אעדיה ת.ז. 322618075

המנחה: ד״ר עמי האופטמן.

תאריך: 7.7.2022, יום חמישי, סמסטר ב׳.

תוכן העניינים

תקציר

מטרתו של הפרויקט הינה לזהות שגיאות שנוצרו עקב דיגיטציה של מסמכים בערבית באמצעות OCR (optical character recognition). כחלק מן המידע המסופק לצורך הפרויקט, נעשה שימוש במסמכים של חוקים בערבית, שעברו OCR ואת המקור שלהן בפורמט PDF.

הפרויקט נעשה בשיתוף פעולה עם משרד המשפטים.

מושגים

קריאה צמודה: טכניקה בביקורת הספרות, המתארת פרשנות מעמיקה של קטע קצר. קריאה כזו נותנת משקל גדול רב יותר לפרטי על חשבון הכללי. הקריאה הצמודה מקדישה תשומת לב רבה למילים, לתחביר ולסדר שבו נפרשים משפטים ורעיונות בזמן הקריאה.

קישור

מדעי הרוח הדיגיטליים: מדעי הרוח הדיגיטליים (באנגלית: Digital Humanities) הם שדה מחקר, הוראה ויצירה, המצוי בממשק שבין מחשוב לבין תחומי מדעי הרוח השונים. מטרתם של מדעי הרוח הדיגיטליים היא להרחיב ואף להגדיר מחדש את המחקר המסורתי במדעי הרוח, על ידי שימוש בשיטות ובכלים דיגיטליים. קישור

דיגיטציה: בתחום אחסון הנתונים, דִּיגִיטַצְיָה (או דִּיגִיטִיזַצְיָה, באנגלית: Digitizing או Digitization. על פי האקדמיה ללשון העברית: סִפְרוּת[1]) היא יצירת עותק ממוחשב של תכנים שלא היו בפורמט ממוחשב קודם לכן. קישור

OCR: זיהוי תווים אופטי (באנגלית: Optical Character Recognition) היא טכנולוגיה להמרת תמונה, טקסט מודפס וכתב יד שנסרקו על ידי סורק, למסמך תמליל ממוחשב.

קישור

ReadIRIS: תוכנת OCR רבת עוצמה שתוכננה במיוחד עבור משתמשים פרטיים קטנים עד גדולים משתמשים במשרד. קישור

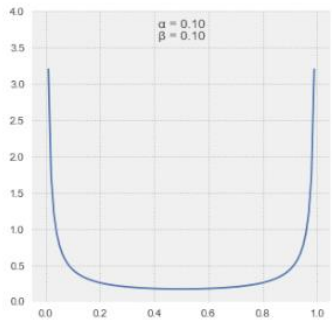
True-PDF: ידועים גם כקובצי PDF מבוססי טקסט או שנוצרו דיגיטלית, קובצי PDF אלה נוצרים באמצעות תוכנות כגון Microsoft Word, Excel. קישור

XML: ראשי תיבות של extensible Markup Language. סוג של קבצים, דומה ל HTML עם אפשרות להגדיר תגיות בעצמנו. שימוש ב־XML מקל על החלפת נתונים בין מערכות שונות שפועלות על גבי תשתיות שונות. קישור

Web scraping: חילוץ נתוני רשת, משמש לחילוץ נתונים מאתרים. תוכנת web scraping עשויה לגשת ישירות לרשת העולמית באמצעות פרוטוקול העברת Hypertext או דפדפן אינטרנט. קישור

התפלגות בטא: משפחה של התפלגויות רציפות, המוגדרות על הקטע [0,1] ובעלות שני פרמטרים המשפיעים על צורת ההתפלגות: α ו-β. השתמשנו בה כדי לבחור שגיאות מהמלונים שהכנו בצורה חכמה.

$$\frac{(1-x)^{\beta-1}\, x^{\alpha-1}}{B(\alpha,\ \beta)}$$



רקע ומטרה

מטרתו של הפרויקט הינו לזהות שגיאות שנוצרו עקב דיגיטציה של מסמכים באמצעות OCR. לשם כך נעשה שימוש במסמכים 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.

דוגמאות

להלן דוגמאות נפוצות שנתקלנו בהן בעת ביצוע המשימה:

1. שגיאות כתיב
   a. החלפה בין ح ל ج: صفحة -> صفجة.
   b. החלפה הין ز ל ذ: يجوز -> يجوذ
   c. החלפה בין ه ל ة: ةأ-> هأ
2. בלבול בסימנו פיסוק
   a. נקודה (.) –> לפסיק (,).
   b. פסיק (,) -> לגרש עליון (').

תיאור המערכת

- קריאה ומיפוי לפי מזהה חוק של קבצי ה XML , הוצאת הטקסט מהקבצים הנ"ל.
- שאיבת מידע נוסף, כ 1990 חוקים נוספים מאתר ויקיטקטס.
- ביצוע עיבוד מקדים רלוונטי על הטקסט בעזרת ביטויים רגולריים.
- מעבר על כ 100 מסמכים באמצעות קריאה צמודה על מנת להכין מילונים של שגיאות נפוצות וכמות הופעתן.
- פגימה מכוונת בטקסט החוקים באמצעות המילונים שהכנו.
- חלוקת המידע לסטים ויצירת טוקניזר הממפה מילים למספרים. הכנסנו את כלל המילים שנמצאות בסט האימון.
- הזנת המידע למודל שאומן מראש וחיזוי למילים שמכילות שגיאות. לבסוף בחינת התוצאות ובחירת המודל הטוב ביותר.

אמצעים טכנולוגיים

המערכת ממומשת בשפת Python גרסה 3.8, סביבת המחקר היא Jupiter Notebook וסביבת הפיתוח הינה VS-Code. לוודא שיש פייטון 3.8 ומעלה במחשב. מומלץ להשתמש ב-Anaconda שניתן להוריד בקישור הבא:

https://www.anaconda.com/products/individual

בסיס הנתונים: SQLite.

סט הנתונים הוא סט של טקסטים - מסמכים שעברו OCR בפורמט text. יש כ 9000 טקסטים בפורמט DOC ואת המקור שלהם בפורמט PDF.

טכנולוגיות נוספות וספריות:

Numpy – ספריית קוד שמספקת תמיכה במערכים גדולים דו ממדיים ובמטריצות, ומבחר של פעולות מתמטיות שניתן לשלב איתם

Argparse – מאפשרת לעבוד עם הקוד דרך העברת ארגומנטים בשורת הפקודות

Pandas – ספרייה שמיועדת לניתוח ועיבוד נתונים.

Matplotlib – ספרייה שנשתמש בה להצגת הנתונים באמצעות תרשימים.

Searborn – ספרייה שנשתמש בה להצגת הנתונים באמצעות תרשימים, בנוסף לספריה הקודמת (שתי הספריות עובדות ביחד).

Sklearn – לייבוא אלגוריתמים של למידת מכונה ולשימוש במטריקות על מנת לבחון את ההתקדמות בזמן האימון וגם בפונקציית הפסד כדי שהרשת תוכל להתאים את המשקולות.

Tensorflow 2.0 - לבנייה ואימון רשת נוירונים.

Keras – לשימוש ב callbacks להשגחה בזמן האימון כדי לדעת איך המודל עובד ולשפר אותו, וכדי להיעזר ב keras.layers להגדרת הרשת

BeautifulSoup – השתמשנו בה כדי לשאוב טקסטים מאתר WIKITEXT.

מוטיבציה

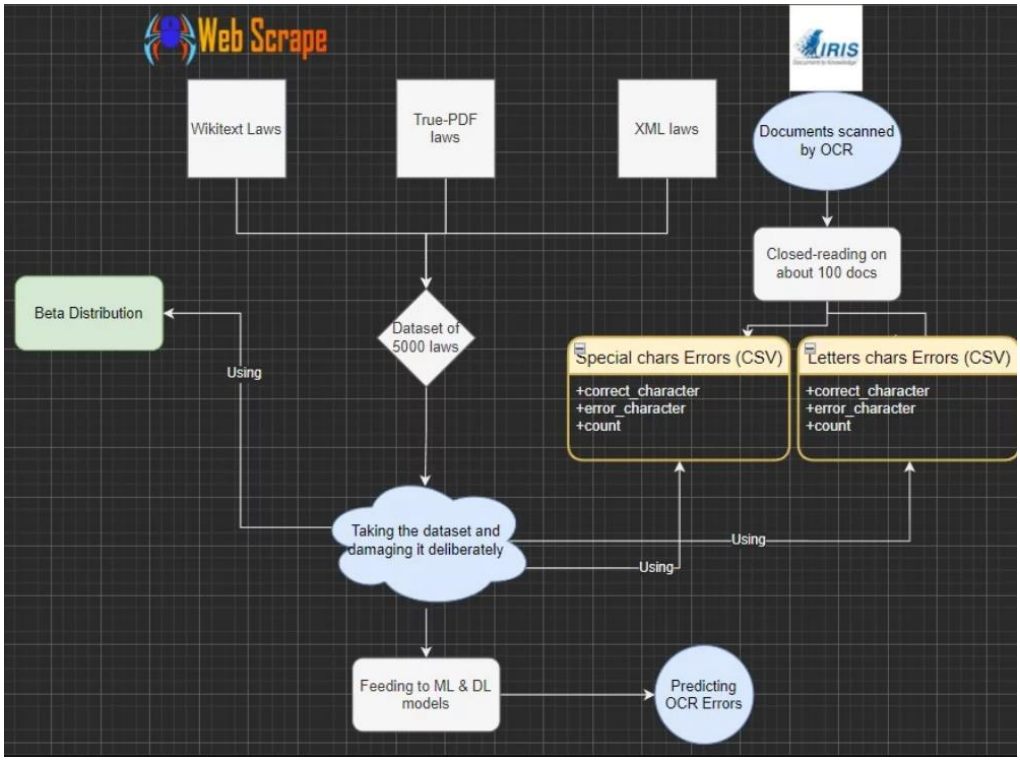
מבין המטרות העיקריות שהפרויקט הזה עוזר להשיג בהמשך:

1. הדרך היחידה בה החוק זמין בערבית זה רק אחרי שעושים דיגיטציה של כל החוברות - ואז על ידי אלגוריתם נוסף - לסכם מה חוק שהוא קיים היום ומה ישן והוחלף כבר, וזה רק אחרי דיגיטציה.
2. חשוב למחקר על התפתחות המשפט הישראלי.
3. חלק גדול מהחוקים הישנים שעבדו איתם עדיין בתוקף (כמו שיש עוד חוקים מהמנדט הבריטי ואפילו מהתורכים שעוד בתוקף).
4. יש חוקים שגם אם הם עכשיו כבר לא בתוקף, הם היו בתוקף כשקרו דברים בעבר ולכן זה חשוב לאנשים להגן על הזכויות שלהם.
5. אם כל החוק יהיה נגיש בערבית - אזרחים שרק מדברים ערבית יוכלו לדעת מה החוק
6. המטרה היא שכל החוקים בישראל יהיו נגישים. זה יאפשר בשלב הבא למשל לאמן מנוע תרגום משפטי - מעברית לערבית ולהיפך, או מתרגם לטפסים משפטיים (אבל זה פרויקט אחר).

אתגרים שהיו לנו בהמשך הפרויקט

1. המסמכים לא כולם באותה איכות.
2. יש מסמכים לא קריאים גם לבני אדם, היינו צריכים לעבור על המסמכים ולהוציא מסמכים כאלה וזה תהליך שלקח זמן מאתנו.

מבנה המערכת



הסבר על רכיבי המערכת:

1. יש 3 מקורות לסט הנתונים:
   a. אתר WIKITEXT (בעזרת scraping).
   b. קבצי true pdf בערבית.
   c. קבצי XML חצי מובנים.
2. קבצי מקור ב PDF שעברו סריקה בעזרת תוכנת READIRIS.
3. בניית שני מלונים של שגיאות אחרי ביצוע קריאה צמודה על כ 100 מסמכים סרוקים.

4. שימוש בהתפלגות בטא על מנת לבצע פגימה מכוונת על סט הנתונים (ראה בדף המושגים: **התפלגות ביטא**).
5. חיזוי השגיאות בעזרת שיטות של למידה עמוקה.

תיאור הפרויקט

על מנת ללמד את המודל שלנו נעזרנו במסמכים חצי מובנים, כלומר שימוש קבצי ה XML וכן חילוץ מהם את תוכן התגיות הרלוונטית (חילוץ הטקסט מהתוויות). בנוסף נעזרנו במסמכים לא מובנים (קבצי ה DOCX), כלומר תוצרי מסמכי ה OCR 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 בביטויים רגולריים על מנת לנקות את הטקסט ולהכין אותו לשלב הטוקניזציה.

ביצענו קריאה צמודה על כ100 מסמכים שעברו OCR על מנת להכין מילונים של שגיאות נפוצות.

נעזרנו ברשת נוירונים שמטרתה ללמוד את ההקשרים בין המילים וכן מנסה לתת משקל דומה למילים אשר קרובות במשמעות הסמנטית או באות בד"כ אחת אחרי השנייה, נעזרנו בתיוג בינארי על מנת לתייג את המילים החשודות לשגיאות.

פתרון הבעיה

כעת נסביר על השלבים של תהליך העבודה שלנו וכיצד הגענו לתוצאות שאותן נפרט בהמשך.

שלבים מקדימים

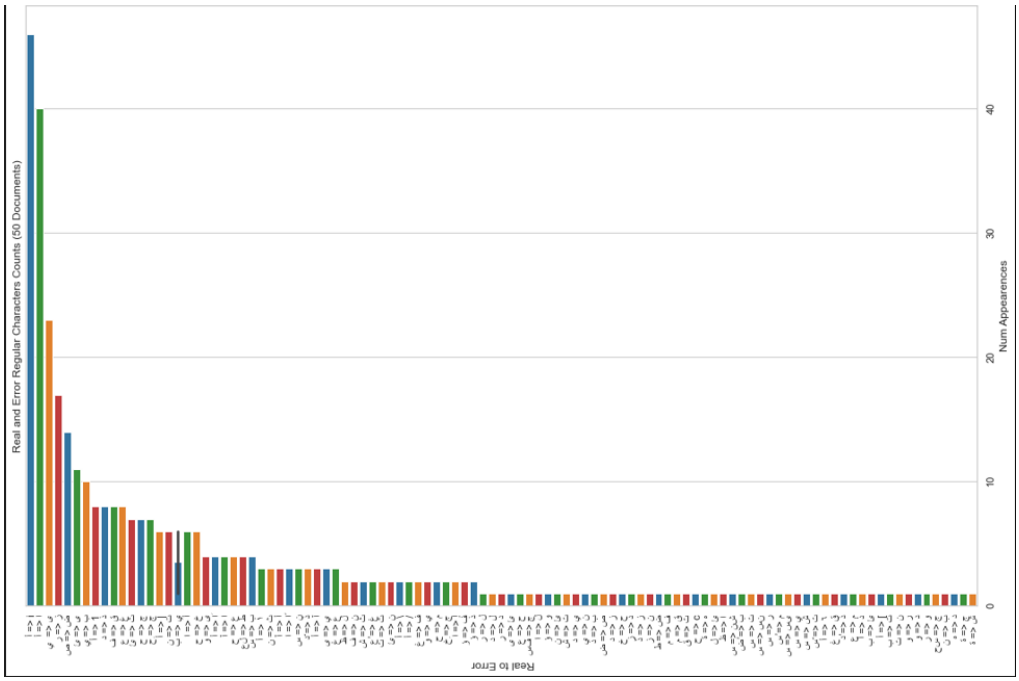
1. קריאה ומיפוי לפי מזהה חוק של קבצי ה,XML הוצאת הטקסט מהקבצים הנ"ל. בשלב זה לקחנו מסמכים שקיבלנו ממשרד המשפטים בפורמט XML. לכל מסמך יש: מזהה ייחודי, כותרת ותוכן. בנינו dataframe שמכיל את כל המסמכים עם המזהה והכותרת שלו, נעזרנו ב DOM Manipulation כדי למפות כל מסמך עם המזהה שלו.
2. שאיבת מידע נוסף, כ2000 חוקים נוספים מאתר ויקיטקטס. בעזרת DOM Manipulation הכנו dataframe נוסף שזהה מבחינת המבנה שלו ל dataframe הקודם עבור מסמכים של חוקים ששאבנו מאתר web.

3. ביצוע עיבוד מקדים רלוונטי על הטקסט. לדוג׳ הוספת רווחים בין מירכאות במקומות שצריך, החלפת מירכאות מיוחדות במירכאות רגילות, חילוץ סימנים בין סוגריים מרובעים ועוד... נעזרנו בביטויים רגולריים כדי להגדיר דפוסים שמנקים את הטקסט, דוגמה:

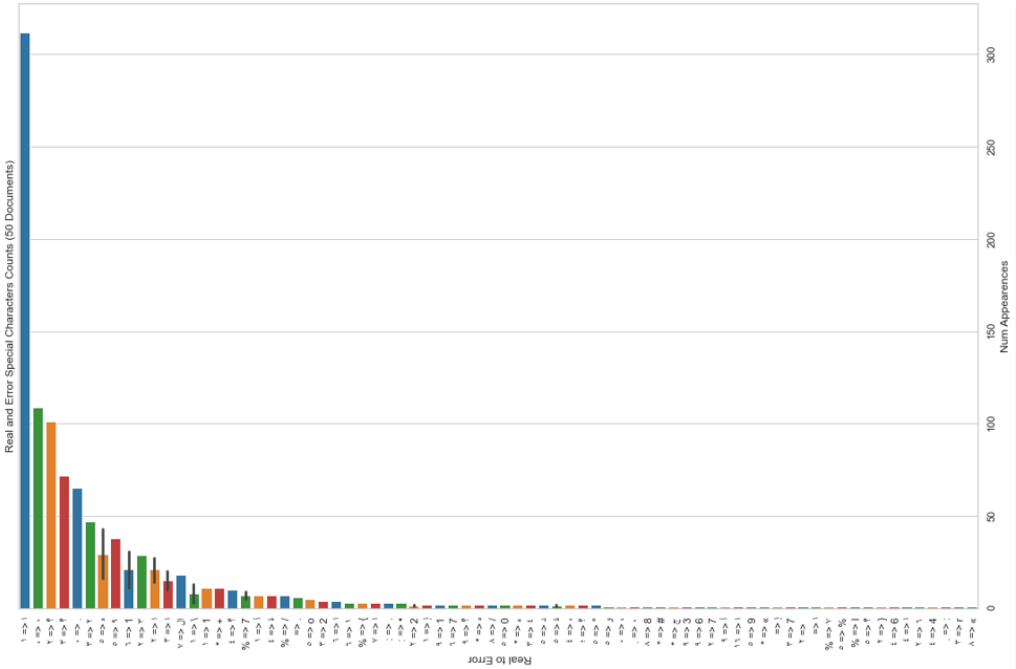
```
remove extra new lines
data = re.sub(r'\n\s*\n', '\n', data)
```

4. מעבר על כ100 מסמכים באמצעות קריאה צמודה על מנת להכין מילונים של שגיאות נפוצות וכמות הופעתן. ניתן לראות דוגמה להיסטוגרמה בעמוד הבא.

התפלגות השגיאות של אותיות בשפה:



התפלגות השגיאות של תווים מיוחדים:



5. פגימה מכוונת בטקסט החוקים באמצעות המילונים שהכנו (פירוט על פונקציית הפגימה בהמשך) חלוקת המידע ל3 סטים: אימון, מבחן וולידציה, כמו כן התאמתו לצורה אחידה לצורך הכנסתו למודלים. עוברים על כל טקסט מסט הנתונים שהכנו (כ 5000 מסמכים) ופגמנו כ 20% מכל מסמך.
6. יצירת טוקניזר הממפה מילים למספרים, תהליך שבו מכינים כל מילה להיכנס למודל למידה עמוקה בהמשך.
7. הכנת מודל למידה עמוקה, שימוש במטריקות על מנת לבחון את ההתקדמות בזמן האימון, הכנת פונקציית הפסד על מנת לבצע אופטימיזציה למודל. פונקציית ההפסד שהשתמשנו בה היא binary cross entropy שמענישה את הרשת במשקל מסוים אם החיזוי אינו נכון (False Positive או False Negative). אנחנו רוצים לנסות עוד פונקציות הפסד בתקווה שזה ישפר את התוצאות. המודל של הלמידה העמוקה שהשתמשנו בו הוא LSTM, זאת גרסה מיוחדת של RNN שהארכיטקטורה שלה מאפשרת למודל להבין את ההקשר בין המונחים בטקסט. הבנת ההקשר היא קריטית כדי לזהות איזו מילה מכילה שגיאה ואיזו מילה לא.
8. בחינת התוצאות, בחירת המודל הטוב ביותר ושמירת המשקולות שלו.

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 בטא. בנינו התפלגות בטא על פי כמות המופעים של כל טעות וכל פעם דגמנו אחת. התוצאה היא שתחילה נדגום טעויות יותר נפוצות ולאט לאט נתקדם לפחות. זה הבטיח רנדומליות אבל עם זאת גם נאמנות לסוג מידע שאנחנו מעוניינים לחזות עליו. בנוסף קבענו שנפגום ב20% מכל מסמך כיוון שלאחר בחינת 100 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. לדוגמה:

על המודל לחזות עבור כל מסמך את המטריצה הנ"ל עבור כל מסמך.

התאמת המידע לצורה אחידה

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 ו,200 כלומר כל

משפט בסט האימון יכול להכיל 50 מילים (מופרדות ברווחים) ויכולים להיות בו 200 משפטים (את היתר מרפדים באפסים).

בחלק של החיזוי על מסמכי ה docx, 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.

שלבי התוכנית העיקרית

מטרת השלבים הללו היא להשתמש במודל שאומן לצורך חיזוי השגיאות.

1. קבלת נתיב לתיקייה המכילה קבצי docx וקריאתם.
2. ביצוע עיבוד מקדים (כפי שבוצע קודם) על הטקסט של קבצי ה docx.
3. המרת המידע לגדלים קבועים וביצוע טוקניזציה ואז שמירת מטריצה אשר תאפשר מיפוי חזרה לגדלים המקוריים של הקבצים.
4. הזנת המידע למודל שאומן מראש וחיזוי שחשודות שמכילות שגיאות.

5. סימון המילים החשודות לפי חיזוי המודל, שימוש במטריצת המיפוי לצורך סימון כלל הקבצים בצורה נכונה.
6. שמירת הקבצים המסומנים בפורמט txt בתיקייה ייעודית.

זה המדדים שקיבלנו בסוף האימון שהיה יחסית קצר:

True positive = 127473

True negative = 28207033

False positive = 40665

False negative = 4829

```
Measurements:
- Accuracy: 90%
- Precision: 76%
- Recall: 96%
```

מצב קיים ודוגמאות חיזוי

	Original	Prediction
Example 1	رئيس الكنيست رؤبين	رئيس الكنيست <e>رؤبين<e>
Example 2	المادة ١ - تلغى	المادة ١ - تلغى

لدوغمة, مه شمسومن باءدوم زه
ميلم لا تقينم, وهوا הצליח
لاءتر ولهتريع عليهن.

:_____

ميله راءشونه زه شم של נשיא
המדינה לשעבר (راءوبن) ويش به
טעות באות השלישית

مه شبءذ يمين زه مسفر عم אות
(אליף) שהפך ל i באנגלית.

האחרונה היא נקודה שהפכה
לספרה.

:_____

مه شبכחول זה מילה נכונה
שהוא סימן בטעות.

```
. لدا القانون
<e>شمعون<e> بير رئيس الدولة
بنيامين نتنياهو رئيس الحكومة
يعقوب نئيمان وزير العدل
ريفلين رئيس الكنيست <e>رؤيين<e>
لسنة ٥٧٧٠ - ٢٠٠٩ المادة ١ - بعد المادة
تضع / <e>i٧٠<e> مركبة المادة <e>"يقاف<e>
. ٥ . ٠ < ٠<e><e> محمة
```

```
الغاء
المادة ١٧٤
<e>اضافة<e> المادة <e>أ٣٨٤<e>
الغاء
المادة أ٣٩٣
تعديل المادة ٤٥٣
٢٢
٢٠٠٩ - ٥٧٧٠ لسنة ( تعديل رقم ١٠١ ) قانون العقوبات.
( القانون الأصلي - فيما يلي ) ١١٩٧٧ - ٥٧٣٧ لسنة العقوبات قانون من ١٧٤ المادة تلغى - ١ المادة .
يحل الأصلي القانون من ٣٨٤ المادة بعد - ٢ المادة :
<e>سة<e> المادة <e>i٣٨٤<e> - ( ١ ) سرق من كل سنوات أربع مدة بالحبس يعاقب
```

גם כאן הוא הצליח לחזות את
רוב השגיאות, אבל גם טעה
קצת, לדוגמה סימן מספר
כשגיאה כשהוא לא. (שורה
שלישית הכי ימינה), כל השאר
הן אכן שגיאות.

בנוסף, שורה שלישית מלמטה
(שורה הכי ארוכה בתמונה) בצד
ימין, יש שגיאה שהוא לא מצא
(האות ا הייתה צריכה להיות
ספרה 1)

הצעות לשיפור

- ביצוע עיבוד מקדים יותר מקיף – ניתן למצוא עוד anomalies במסמכים ולהוסיף אותם לצורך חיזוי יותר טוב.
- שימוש במודל שאומן מראש, כמו BERT ולהתאים אותו לבעיה שלנו. ראשי תיבות של transformers bidirectional encoder representations, נועד לבצע ניתוח של שפה טבעית NLP (natural language processing) ולבצע סיווג וניתוח סמנטי.
- הוספת עוד מידע חיצוני על מנת שהמודל ילמד יותר את ההקשרים.
- מעבר על עוד מסמכים ומציאות שגיאות נפוצות.

שימוש במערכת

1. העתקת הנתיב שמכיל את הקבצים שרוצים למצוא את השגיאות בתוכם.
2. פתיחת טרמינל והרצת הפקודה:

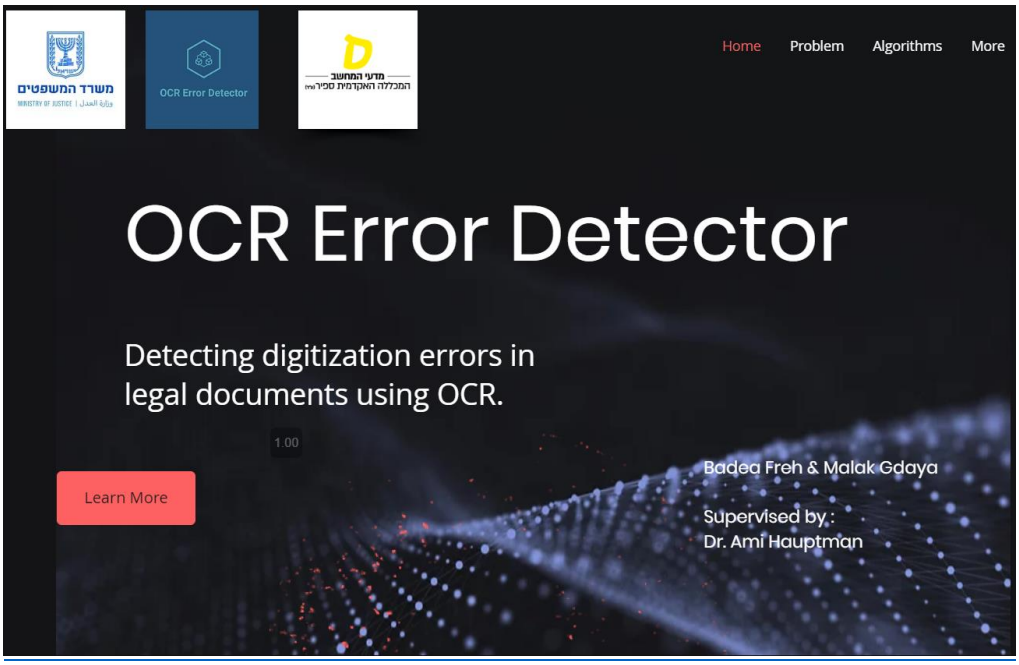
```
python script.py docx_path
```

עבודה עתידית

- בהמשך אנחנו רוצים שהמערכת תעבוד גם על דפוס כתיבה שונה (עלולים להופיע סוגים אחרים של שגיאות).
- לשפר את הביצועים של המערכת ולקבל דיוק יותר גבוה.

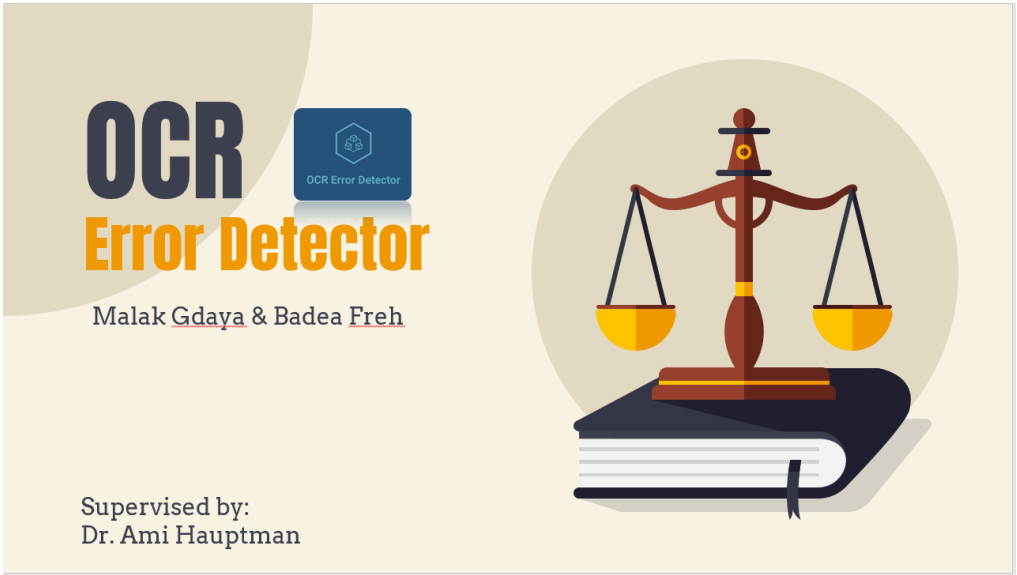
נספחים

קישור לפוסטר: https://digitization1.wixsite.com/ocr-error-detector



קישור למצגת:

https://docs.google.com/presentation/d/1NBj5rTykxU_8fI9kHfeb4SxmH7vYj6km/edit?usp=sharing&ouid=108788761315487159050&rtpof=true&sd=true



ביבליוגרפיה

באנגלית:

- [Guide to LSTM Model] https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21
- [Legislative XML for the Semantic Web] https://cadmus.eui.eu/handle/1814/18715
- [XML Format] https://www.legislation.gov.uk/developer/formats/xml
- [An XML Standard for Legislation in the Netherlands: Primer] https://www.researchgate.net/publication/2538576_An_XML_Standard_for_Legislation_in_the_Netherlands_Primer

בעברית:

- [רמת העברית של התלמידים הערבים בירידה – וגם הסיכוי שלהם להרוויח כמו יהודים] https://www.haaretz.co.il/news/education/2020-03-09/ty-article-magazine/.premium/0000017f-e378-d38f-a57f-e77a5bc40000
- [התוכנית לחיזוק העברית במגזר הערבי חורקת: אין שיפור ברמת התלמידים] https://www.ynet.co.il/articles/0,7340,L-5446661,00.html