**תיעוד**

* לא לשכוח שבצענו Project characterization

חומרי קריאה:

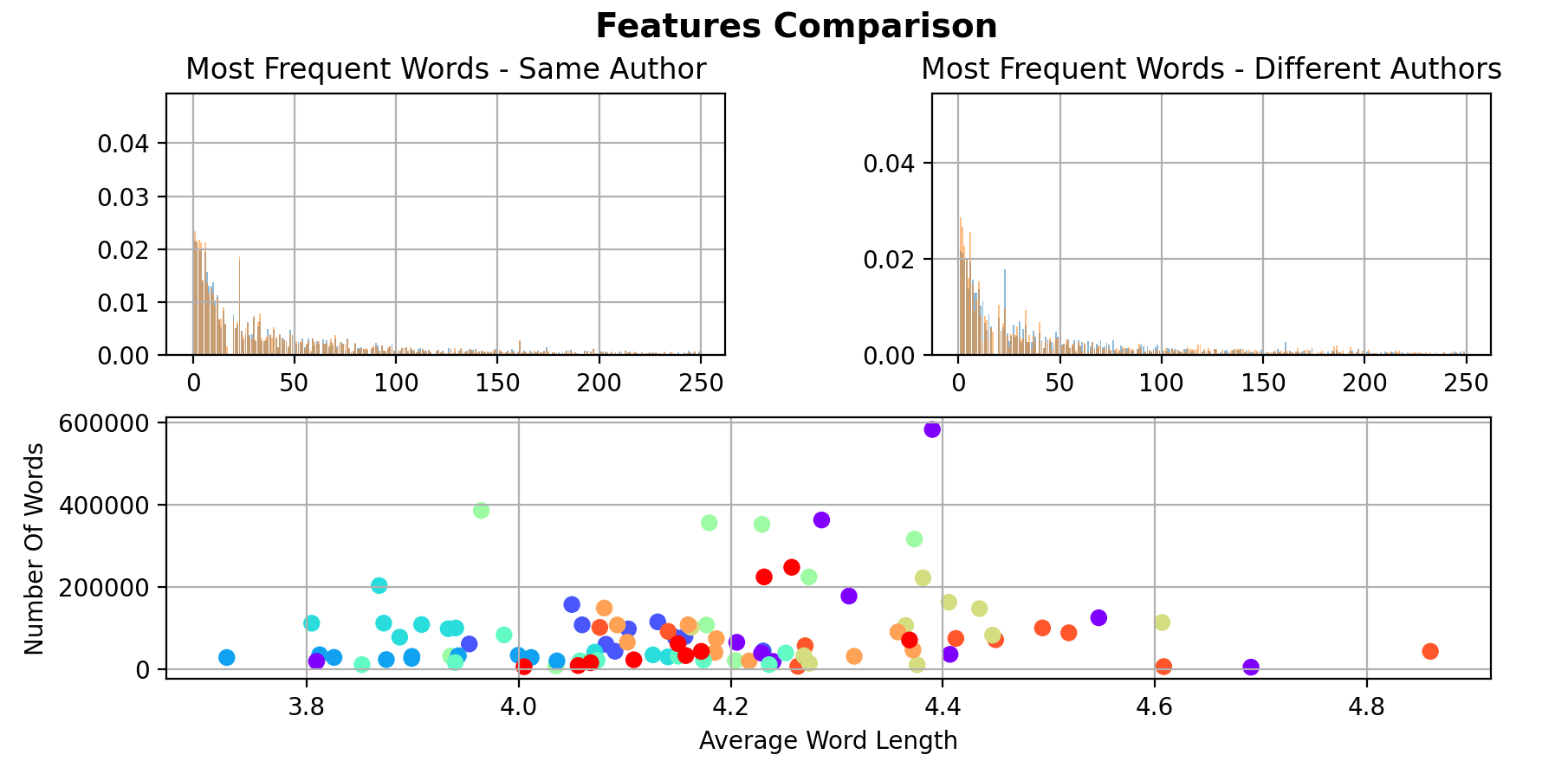
לקחת מהרפרנסים של המצגת אמצע

* [http://karpathy.github.io/2015/05/21/rnn-effectiveness/](https://eur01.safelinks.protection.outlook.com/?url=http%3A%2F%2Fkarpathy.github.io%2F2015%2F05%2F21%2Frnn-effectiveness%2F&data=02%7C01%7Ckiassi%40campus.technion.ac.il%7C6f096115164e4e6887dc08d85ad18f0f%7Cf1502c4cee2e411c9715c855f6753b84%7C1%7C0%7C637359203385747454&sdata=RdvMHVLj%2FnJj%2Bm6nBpX5lMUNzUXvoxuQom%2BBrV%2B3Yv8%3D&reserved=0)
* [https://www.tensorflow.org/tutorials/text/text\_generation](https://eur01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fwww.tensorflow.org%2Ftutorials%2Ftext%2Ftext_generation&data=02%7C01%7Ckiassi%40campus.technion.ac.il%7C6f096115164e4e6887dc08d85ad18f0f%7Cf1502c4cee2e411c9715c855f6753b84%7C1%7C0%7C637359203385747454&sdata=71oOqiFirVlavN5Hi5Ag9k0RLatRWJh4fWklEYNppCE%3D&reserved=0)
* [https://machinelearningmastery.com/how-to-develop-a-word-level-neural-language-model-in-keras/](https://eur01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fmachinelearningmastery.com%2Fhow-to-develop-a-word-level-neural-language-model-in-keras%2F&data=02%7C01%7Ckiassi%40campus.technion.ac.il%7C6f096115164e4e6887dc08d85ad18f0f%7Cf1502c4cee2e411c9715c855f6753b84%7C1%7C0%7C637359203385757447&sdata=ip%2BzON%2BMnZlEykBbrgXIoIqmD5QH9KldJiLFrMkqiPk%3D&reserved=0)
* <https://machinetalk.org/2019/02/08/text-generation-with-pytorch/>
* <https://www.kdnuggets.com/2020/07/pytorch-lstm-text-generation-tutorial.html>

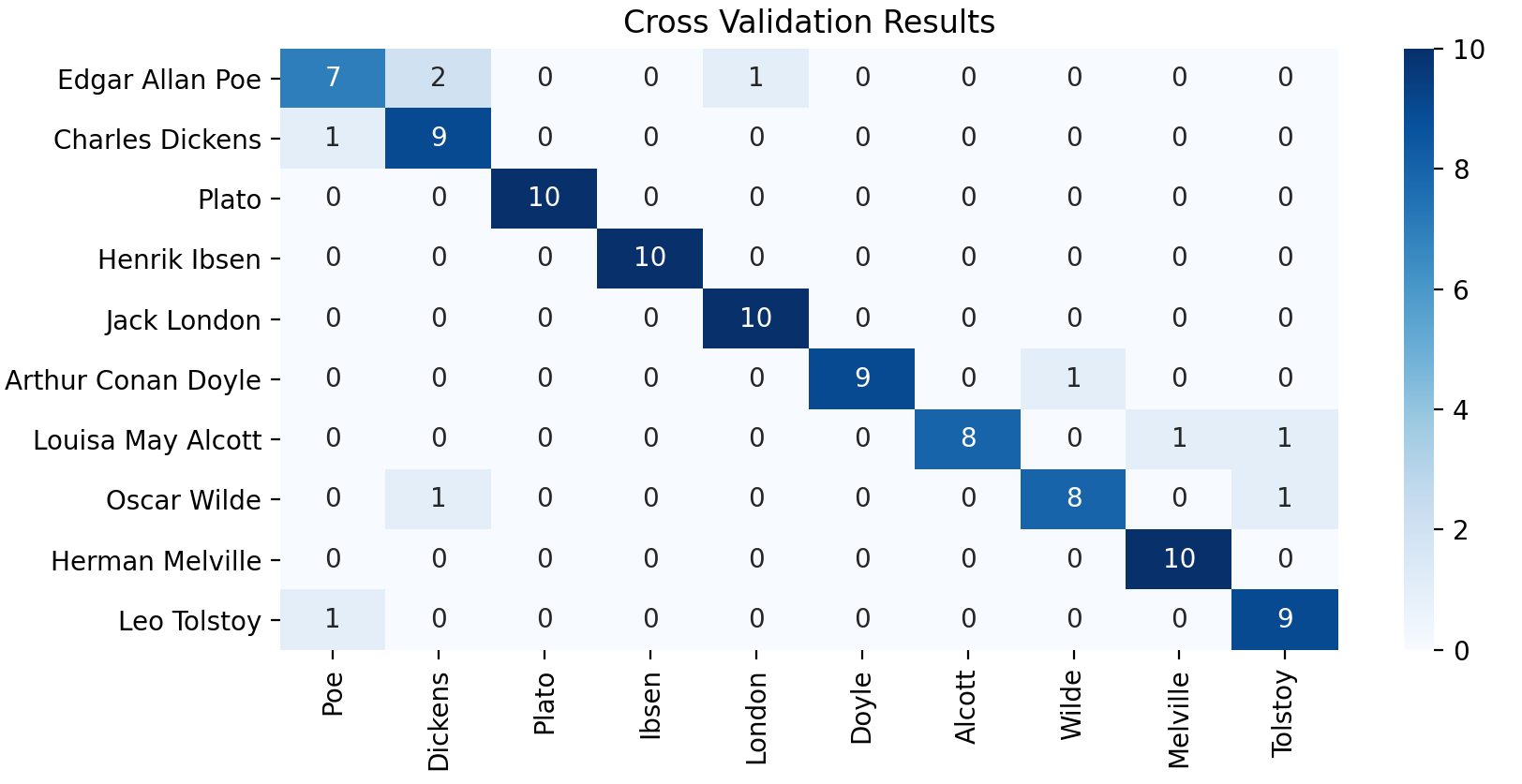
טוי פרובלם (הכל מופיע במצגת אמצע):

* משימת זיהוי כותבים פשוטה.
* דאטהסט קטן: 10 כותבים שלוקחים מכל אחד מהם 10 ספרים שכתב (באמצעות פרויקט Guttenberg). פיצול של 85/15% של טריינינג וטסט.
* השתמשנו במודל Scikit-learns SVC .
* מבחינת preprocessing:
  + “Cleaning” the texts
  + Tokenizing
  + Ignoring punctuation
* מבחינת פיצ'רים:
* Histogram of most frequent words
* Histogram of least frequent words
* Average word length
* A picture containing clock, meter, kitchen

  Description automatically generatedNumber of words

A screenshot of a cell phone

Description automatically generated

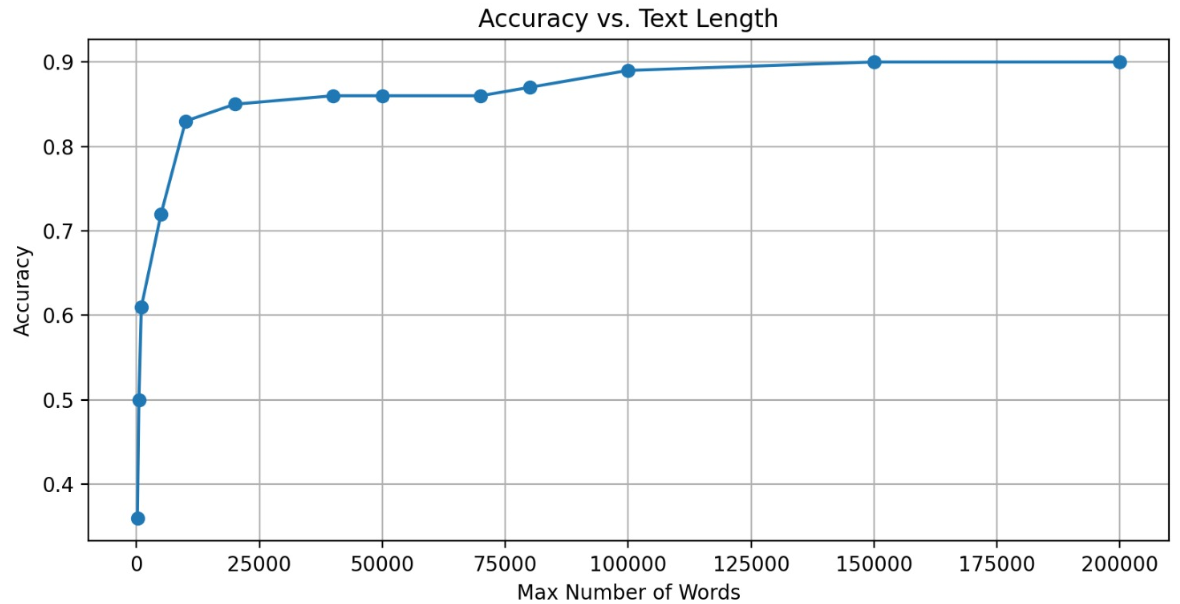
תוצאות:

דיוק ממוצע: 90%

על הפרויקט (גם במצגת אמצע):

* מטרה
* מוטיבציה
* (רקע על NLP ו-Authorship Attribution)
* אתגרים
* סקר ספרות
* סביבת עבודה - Python 3, Anaconda, pycharm, nltk ועוד.

בין לבין, בצענו בדיקה על הטוי פרובלם לבדוק דיוק למול אורך טקסט (לקחנו כל פעם מספר מסוים של מילים מטקסט ובדקנו):



כלומר, קיימת השפעה משמעותית של אורך הטקסט על הדיוק.

* הכנת ה-data בפרויקט (לפרט על כל פרסור: html לטקסט, פידיאף לטקסט, ניקיונות שעשינו, לחפור על זה שהשתמשנו בregex, הקודש כתבנו בפייתון, ניסיונות שעשינו ב-CMD, סידור ידני שעשינו לעמודות בפידיאף ועוד):
* BMJ:
  + הורדה של 190-200 reviews עם תיוג, עבור 9 כותבים, כדאטה לטסט.
* הורדה של 10 מאמרים לכל כותב כדאטה לאימון (הכותב לכל הפחות כותב מרכזי במאמר).
* עבודה ידנית של ניקוי ה-reviews (הורדה של דפי html וניקויים) והמאמרים (חיבור ידני של שתי עמודות לאחת מקבצי pdf, הרבה preprocessing שיפורט בהמשך ועוד).
* בינתיים: זה הדאטה איתו מתעסקים בפרויקט.
* Nips:
* הורדנו ופרסרנו (כולל ניקיון משמעותי של הטקסט) X reviews.
* F1000:
  + הורדנו מעל ל11 אלף reviews ופרסרנו אותם לקבצי טקסט נקיים.
* מאמרים של מרק:
  + הורדנו את המאמרים כpdfים בלבד.

Preprocessing:

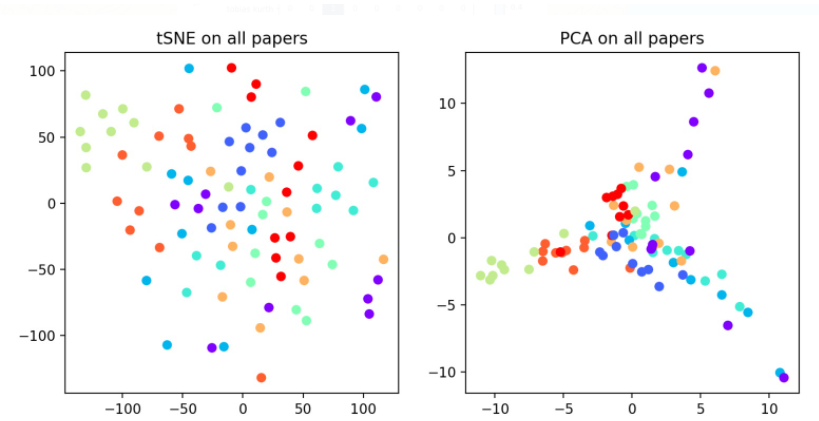
* ניקוי ידני של מידע לא רלוונטי ואף של מידע "מפליל" – שם הכותב, כתובת המייל שלו וכו'.
* עבודה ידנית של איחוד שתי עמודות לאחת.
* התעלמות מחלק מהניקוד.
* התעלמות מ-stopwords.
* Stemming.
* Lemmatization.

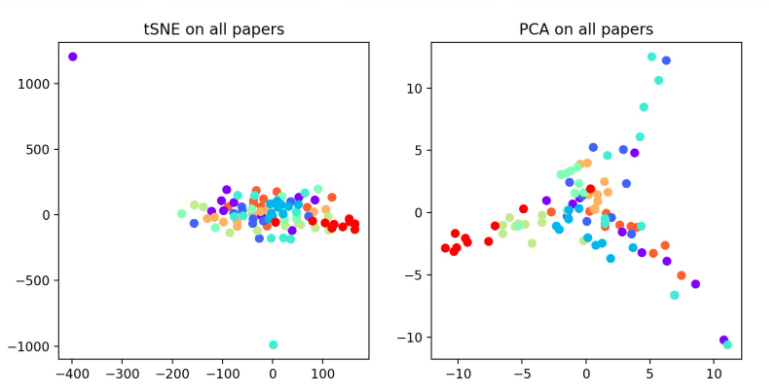
Features:

* היסטוגרמת 1-gram.
* היסטוגרמת 2-gram.
* היסטוגרמת 3-gram.
* TF-IDF.
* קלאסטרינג (בין כל המאמרים, בין מאמרים של אותו כותב, בין reviews(?)):
* מוטיבציה.
* שיטות שונות (PCA, tsne, dbscan).
* ניסוי וטעייה: שינויים שונים בפיצ'רים ובדיקת האשכול, שינויים שונים ב-preprocessing ובדיקה מתאימה.

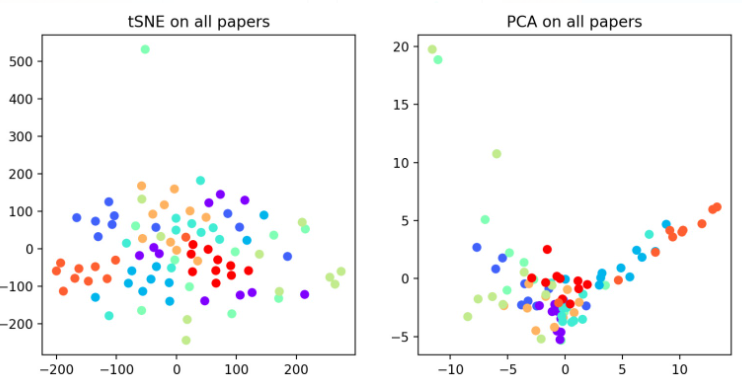
דוגמאות (בכל גרף – צבע זהה אומר שמדובר באותו כותב מאמר):

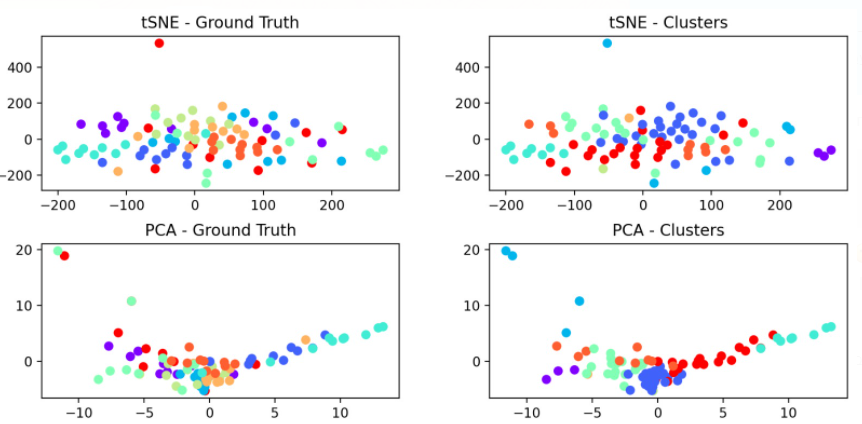
עם lemmatization + הורדת stopwords:



בלי ';' :

עם 4-grams ו- 5-grams:



ובאופן כללי אשכול של המודל הנוכחי (אין קשר בין הצבעים בין גרפים שונים):

* בדיקה על בעיה פשוטה יותר של אימון על reviews במקום על מאמרים – ורק אז ממשיכים ל-cross domain.
* יצירת Language Models:
  + למדנו מה זה מודל שפה ואיך יוצרים את זה, וגם מה זה LSTM.
  + המטרה – ליצור לכל כותב LM בהתבסס על הטקסטים שהוא כתב (מאמרים). להשתמש בזה כדי לחשב perplexity של טקסט מסוים לפי כל LM ולהכניס את זה כ-feature.
  + הרעיון – ה-perplexity יהיה הכי קטן עבור המודל של הכותב שהכי מתאים לטקסט הזה.
  + מה עשינו:
  + מודול vocabulary: שומר מילון שממיר ממילה למספר עבור כותב מסוים. משתמשים בזה כדי להפוך טקסט שכתוב כמילים לאות מספרי.
  + מודול LanguageModelNN: מודל שפה שמורכב משכבת embedding, שכבת LSTM, ושכבת FC (אולי ישתנה בהמשך). מממש את הרשת שמגדירה את ה-LM, כולל פונקציית forward.
  + מודול dataset שמאפשר לטעון בצורה נוחה את הטקסטים כ-batches של tokens.
  + פונקציית אימון ופונקציה שמגנרטת משפטים לפי ה-LM (בעיקר כדי לבדוק שהמודל למד בצורה הגיונית).
  + מודול LanguageModel שמאחד הכל (LanguageModelNN + vocabulary) ומאפשר שימוש נוח כדי להוציא features: מכיל פונקציית שמירה וטעינה של המודל + פונקציית חישוב perplexity – עבור טקסט מסוים מזינים את הטקסט לרשת ב-sliding window, ומקבלים וקטור הסתבורויות למילה הבאה, עבור המילה הבאה האמיתית לוקחים את ההסתברות וככה מחשבים perplexity על כל הטקסט. הנוסחא:
  + הכנסנו את המודול הזה כחלק מהמודול של Feature extraction והוצאנו פיצ'רים.