**דו"ח מסכם: Yelper**

**פרטים מזהים:**  
עילי פרידמן 305709248  
סער גוטמן 201543808  
אורן שור 204053076

**הגדרת המשימה:**  
שלושתנו מאוד אוהבים אוכל ומסעדות, ואף עבדנו בעבר בעסקים שמוכרים מזון. בימינו ישנם אתרים רבים אשר מספקים פלטפורמה לכתיבת ביקורות על העסקים, ואלו מוכוונים בעיקר כלפי הסועדים. כלומר, כשנרצה לצאת למסעדה מסוימת, נכנס אל האתרים האלו, נראה את הביקורות על המסעדה, ונחליט אם ברצוננו ללכת או לא. אך זהו למעשה מאגר מידע נהדר שבעלי העסקים יכולים לעשות בו שימוש - אם בעלי העסק היו יכולים לקרוא את כל הביקורות, ולהפנים את הביקורת, הם היו יכולים לשפר את ביצועי העסק, ולהגדיל את הרווחים שלהם. אם כבעל עסק הייתי יכול לקרוא את כל הביקורות על מסעדות שדומות לי, ולהבין באילו תחומים הן מצטיינות, ובאילו הן נכשלות, הייתי יכול להתאים את ביצועי העסק שלי לשוק, ולספק ערך מוסף מובהק אל מול המתחרים הישירים שלי.

ברור שלא ריאלי לבעל עסק גדול לקרוא את כל הביקורות שנכתבות עליו, שכן ישנן וודאי מאות ואולי אלפים כאלה. קל וחומר שלא ניתן לבצע השוואה פשוטה לעסקים מתחרים.

על כן, בחרנו ליצור אפליקציה שמאפשרת לבעלי עסקים לראות איזה חלק מהביקורת שנכתבו עליהם חיובי, או שלילי, ובנוסף, להציג לבעל העסק את הנושאים המרכזיים בהם עסקו הביקורות.

בנוסף, הכלי מאפשר לראות את סיכום הביקורות על מסעדות אשר קרובות גיאוגרפית לעסק, או אשר עוסקות בתחום דומה (לדוגמה - מסעדות סיניות).

**דרכי פעולה שונות לפתרון הבעיה:**  
לא מצאנו פתרונות אשר עוסקים באופן ישיר בתחום שבו אנו עוסקים.

ישנה אפליקציה בשום appbot, אשר עוברת על ביקורת בחנות אפליקציות, ומחזירה לבעלי האפליקציה ניתוח על הביקורות שפורסמו. האפליקציה לא מאפשרת ניתוח של אפליקציות דומות.

אפליקציות רבות אשר עוברות על ביקורות ומסמנות ביקורות מזויפות, אך זהו תחום אשר דורש דאטה מסווג מראש.

ישנן שלוש בעיות עיקריות איתן הפרויקט שלנו מתמודד.

* חילוץ נושאים מתוך הביקורות - Topic Modeling
* חישוב דמיון בין קטגוריות - Similarity calculation
* קביעה עבור כל ביקורת האם היא חיובית או שלילית - Sentiment Analysis.

**Top Modeling:**

בתחום ה-Topic Modeling מודל עיקרי הוא LDA. המודל למעשה לומד שתי טבלאות - טבלה אחת שמגדירה, בהינתן נושא, את ההסתברות של כל מילה להשתייך לנושא. הטבלה השנייה מגדירה, בהינתן מסמך מתוך הקורפוס, את הסיכוי של כל נושא להיות במסמך.

מודל זה, בסופו של דבר, מגדיר עבור כל מסמך אוסף של נושאים, ואת ההסתברות של הנושאים להיות חלק מהמסמך. כלומר, עבור כל מסמך נקבל כמה נושאים.

דרך נוספת לבצע Topic Modeling היא ע"י ייצוג המסמכים כווקטור, ולאחר מכן ביצוע Clustering. בשיטה זו נקבל עבור כל מסמך נושא, או Cluster, אחד בלבד.

לאור העובדה שאנחנו עוסקים בביקורות של מסעדות, שמטיבן הן טקסטים קצרים יחסית, החלטנו לעבוד בדרך זו.

**Similarity Calculation:**

בתחום זה, אנו מבצעים דמיון בין הנתונים אשר קיימים לעסק הקיים אל מול טבלת המידע הקיימת. החלטנו לבצע זאת בשני מישורים:  
1. ע"פ דמיון בין נתוני הקטגוריות של בית העסק: בעבור כל בית עסק נתון בקורפוס רשימה אשר מסווגת אותו לקטגוריות שונות. בהינתן בית עסק – בדקנו בעבור כל קטגוריה שקיימת אצלו ברשימה את הדמיון הקיים בעבור כל אחת מן הקטגוריות הנמצאות ברשימות של בתי העסק האחרים. נעזרנו בחבילה FuzzyWuzzy אשר משווה בין סוגי מחרוזות ומחזירה ציון קורלציה בין 0-100. השתמשנו בחישוב מרחק לווינשטיין (Levenshtein Distance) : אשר מגדיר מרחק כמספר המינימלי של פעולות עריכה שיש לבצע על מחרוזת אחת כדי להגיע למחרוזת השנייה, כאשר פעולות העריכה המותרות הן: הוספת אות, מחיקת אות או שינוי אות לאות אחרת. בעבור כל קטגוריה של בית העסק אשר קיבלה ציון דמיון עם הרשימה של בית העסק השני, מצאנו את הערך המקסימלי ואותו שמרנו. לבסוף ריכזנו ממוצע של כלל הקטגוריות וזהו הציון אשר ניתן לדימיון בין 2 בתי עסק נתונים.

2. ע"פ דמיון מבחינת מיקום גיאוגרפי: דמיון נוסף שחשבנו להיעזר בו המיקום הגיאוגרפי של בית העסק ביחס לשאר (אשר נתון גם כן כקורדינאטות בקורפוס). כאן נעזרנו בחבילה GeoPy אשר מכילה מספר רב של פונקציות אשר מודדות ומנתחות קורדינאטות, ובפרט בפונקציה distance – אשר מחשבת את המרחק בין 2 נקודות. גם כאן קיבלנו מרחק בקילומטרים ודירגנו את כלל המסעדות בסדר יורד החל מהעסק הקרוב ביותר אליי וכך הלאה

**Sentiment Analysis:**

ישנן דרכים רבות לבצע Sentiment Analysis, אבל בסופו של דבר, יש צורך להוציא פיצ'רים מתוך הטקסט, ולאחר מכן לבצע משימה של Classification. לאחר חקר של הdataset לא הצלחנו למצוא פיצ'רים אינדיקטיביים (לדוגמה - שעת פרסום הביקורת קשורה לסנטימנט, או שישנו קשר בין אורך הביקורת לסנטימנט ועוד), ולכן הסתפקנו בCountVectorizer שבונה ייצוג של המסמך כווקטור. לביצוע הסיווג בחרנו במודל של Logistic Regression.

**הוראות התקנת הכלי:**

הכלים המרכזיים בהם השתמשנו הם:

1. Dask - חבילה אשר מתבססת על Pandas ותומכת בdataframes גדולים מאוד, ומאפשר ביצוע שאילתות באופן יעיל.

2. Scikit-learn - חבילה שהיא חלק מAnaconda, לטובת חלק מהמודלים שפיתחנו.

3. TKinter - לבניית הGUI של האפליקצייה.

שלוש החבילות המרכזיות בהן השתמשנו הן cross platform, ולכן אינן תלויות במערכת ההפעלה.

להרצת הפרויקט נשתמש בAnaconda Interpreter. כמו כן יש להריץ את קובץ GUI.py

**ההיסטוריה של השימוש בכלי:**

**Dask -**   
היא חבילה שתוכננה כדי לתמוך בעבודה עם מאגרי נתונים גדולים במיוחד. החבילה מאפשר עבודה הן על מחשב בודד, והן על קלאסטר של מחשבים רבים. אחת התכונות המרכזיות של הDataframes של Dask היא שהפעולות על הDataframe מתבצעות באופן lazy, רק כאשר הפעולה נדרשת לשם החישוב הבא.

בין הפרוייקטים שנעזרים בב-Dask ניתן למנות את Scikit-learn, וחבילות נוספות.

**Scikit-kearn-**

חבילה מרכזית מאוד בעולם הmachine learning, מאפשרת לפתח מודלים שונים בתחומי classification, regression ו-clustering. קשה לדמיין פרויקט Machine Learning שלא נעזר בחבילה זו.

**TKinter –**

חבילה לעיצוב GUI. החבילה בשימוש מאוד נרחב, ולמעשה משמשת כסטנדרט של השפה בכל הנוגע לGUI

**DataSet:**

חיפשנו dataset שיכיל כמות גדולה של ביקורות, ומצאנו את הdataset של Yelp. הdataset מכיל כ6.5 מליון ביקורות על כ-200,000 עסקים הפזורים ב-10 ערים. הdataset הזה גדול מאוד, ומאפשר לנו לפתח מודלים טובים, ולבצע ניתוח של ביקורת בחתכים לפי עסקם. כמו כן, הdataset מחזיק קטגוריות עבור כל בית עסק, ואת מיקום העסק, מה שמאפשר לנו לחפש עסקים דומים על פי פרמטרים אלו.

נציין כי בdataset הזה ישנם בנוסף גם מאגר של משתמשים ושל תמונות שהועלו, אך מפאת קוצר זמן לא הספקנו להשתמש במידע זה. ניתן היה, לדוגמה, למצוא "משתמש זהב" שכדאי לקרוא את הביקורת שלו. משתמש כזה השאיר ביקורות במסעדות אחרות, ובאופן עקבי הביקורת שלו דמתה לשורה סיכום כלל הביקורות שקיבל העסק. כלומר, הביקורת שלו מהווה ייצוג טוב של דעת הרוב, ולכן נרצה לתת משקל מיוחד לדעתו.

**איך השימוש בכלי בא לידי ביטוי:**

**Dask:**

הdataframe של Dask אפשר את העבודה שלנו עם היקפי המידע הגדולים שהdataset שבחרנו הביא. בניגוד לPandas, הdataframe של Dask לא טוען לזיכרון את כל המידע בזמן בניית מבנה הנתונים, אלא רק כאשר יש צורך בנתונים לטובת ביצוע חישוב. כך יכולנו לעבוד עם קובץ JSON ששוקל מעל 5 ג'יגה, מבלי שייגמר הזיכרון אפילו על מחשב די פרימיטיבי.

בהקשר של העבודה שלנו, השתמשנו בפונקציונליות שמזכירה מאוד את הפונקציונליות הקיימת ב-Pandas, כמו מעבר ומניפולציה של ערכים בעמודה מסוימת.

כמו כן, השתמשנו בפונקציות read\_pqrquet ו-to\_parquet, אשר שומרות\קוראות את הdataframe מהדיסק, באופן יעיל. לאחר כתיבת הdataframe לדיסק בפורמט הזה, נוכל לקרוא חזרה לזיכרון רק את העמודות שנרצה באופן סלקטיבי, וכך נוכל לייעל עוד יותר את ניהול הזיכרון שלנו.

**Sklearn:**

החבילה מספקת את המודלים אותם למדנו במסגרת הפרויקט.

1. TF-IDF:  
   לטובת ה-Topic Modeling, השלב הראשון שביצענו הוא הפיכת המסמכים (ביקורות), לווקטורים. בתחילת הדרך התלבטנו כיצד לייצג את המסמכים כוקטורים. אופציה ראשונה הייתה Doc2Vec, אשר מסייעת בזמני ריצה בזכות העובדה שהיא מצמצמת מאוד את הייצוג הווקטורי, ולכאורה גם עשויה הייתה לסייע במידול של הנושאים עצמם בשל עובדה זו. בפועל, ראינו שלא הצלחנו להגיע לנושאים קוהרנטיים שניתן לחלץ מהם משמעותי כלשהי, ופעמים רבות קיבלנו קלאסטרים דומים מאוד.

בשלב זה התלבטנו בין TFIDFVectorizer ובין CountVectorizer. ההבדל המהותי בין השניים הוא ש TFIDF בודק את חשיבות המילים על פני כל הקורפוס (כלומר, מילה שמופיעה בכל מסמך בקורפוס מספר רב של פעמים אינה מילה אינדיקטיבית), לעומת CountVectorizer אשר פשוט סוכמת את מספר המופעים של כל מילה בתוך מסמך. TFIDF מספק לנו הפרדה טובה יותר בין קלאסטרים, וגורם לכל נושא להיות קוהרנטי יותר.

1. K-Means:  
   לאחר ייצוג הביקורות כווקטורים, נדרשנו לייצר קלאסטרים. למעשה הקלאסטרים מייצגים חלוקה של הביקורות לנושאים שונים. לאחר שניסינו לבצע Hierarchical Clustering על ידי AgglomerativeClustering של sklearn, ראינו שזמן האימון לוקח המון זמן, בשל הצורך לייצר מספר רב של קלאסטרים.

לאחר מכן עברנו לאלגוריתם K-Means, שאמנם לא מייצר קלאסטרים באופן עקבי, אך זמן הריצה שלו נמוך יותר. בשלב זה נדרשנו לבחור k מתאים, אשר יהווה למעשה את מספר הנושאים שקיימים בקורפוס. לטובת כך פיתחנו מספר רב של מודלים, כשאר בכל אחד מהם ערך K שונה, ומצאנו את הערך המתאים K כך שסכום ריבוע המרחקים בתוך קלאסטר יהיה נמוך מצד אחד, אך עדיין הקלאסטרים מייצגים נושאים קוהרנטיים. לבסוף נבחר הערך K=10 כערך האופטימלי.

1. CountVectorizer:  
   לטובת הSentiment Analysis בחרנו להשתמש בCountVectorizer. לכאורה גם כאן היה נכון יותר להשתמש בTFIDFVectorizer, אך בחרנו להשתמש במודל נוסף כדי התרשם מהיכולות שלו.
2. Tkinter:  
   לאחר ייצוג הביקורות כווקטורים בעזרת CountVectorizer, רצינו לייצר סיווג לביקורת - האם היא חיובית או שלילית. בחרנו ב LogisticRegression כclassifier שלנו למשימה זו.

**Tkinter**:

החבילה עצמה בונה GUI בצורה נוחה מאוד. בה אתה יוצר (כבדרך כלל) מחלקה אשר מגדירה ראשית את כל האלמנטים הנמצאים בעמוד ובהמשך יצרנו פונקציות שונות אשר מחברות את כלל הפונקציונאליות אשר תוארה בתחילת הדו"ח לכפתורים והאלמנטים השונים שיש באפליקציה.

**תוצאות:**

TFIDFVectorizer:  
פרמטרים אותם ניסינו במהלך העבודה עם הווקטורייזר: max\_features, stop\_words, preprocessor.

כזכור, לTFIDFVectorizer הגענו לאחר הניסיון לעבוד עם Doc2Vec. מחיפושים וקריאת מחקרים שמצאנו באינטרנט, ראינו שעבור קורפוס דומה לשלנו מבחינת כמות מסמכים ואורך מסמך ממוצע, נכון לעבוד עם ווקטורים באורך 100-300 מימדים. לאחר שלא קיבלנו תוצאות איכותיות מספיק, עברנו ל-TFIDF. ברירת המחדל היא שגודל הווקטור הוא כמספר ה-Terms שבקורפוס. ערך הפרמטר max\_features הוא זה שמייצג את גודל הווקטור המקסימלי למודל. מצד אחד, ככל שנקבל ווקטור גדול יותר, כך התוצאות יהיו טובות יותר (כי הווקטור ייצג באופן מדוייק יותר את המסמך), אך מצד שני ידרוש זיכרון רב יותר, וכמובן זה האימון של מודל הK-Means יגדל, ויהיה אפקטיבי פחות, בגלל The curse of dimensionality. נקודת האיזון שמצאנו היא ווקטורים באורך מקסימאלי של 8000 מימדים.

Stop words:  
רשימת הstopwords ההתחלתית שלנו היא רשימה של חבילת gensim. לאחר מספר הרצות, ראינו שמילים דומננטיות בתוך הקלאסטרים שלנו לא באמת מייצגות נושא משמעותי. לדוגמה, המילים order, eat, food ועוד. לכן עם הזמן הוספנו מילים אלו לרשימת הstopwords כדי לקבל קלאסטרים איכותיים יותר.

Preprocessing - ביצענו preprocessing פשוט יחסית, שכלל בדיקה של stopwords, lemmatization, והסרת Token-ים שגודלם פחות מ-2. נעזרנו בLemmatizer של WordNet. בחרנו שלא להפעיל Stemming על המילים, שכן מטרתינו היא בסוף להציג למשתמש מילים שהוא יוכל להבין ולקשר לנושא מרכזי אחד. Stemming, בניגוד לLemmatization, לא מביא בהכרח את ה-Token למופע מילוני של המילה. לאור בחירה זו, אנו יכולים להציג מספר מילים אשר חוזרות מספר רב של פעמים במסמכים שבתוך קלאסטר, ולקבל ייצוג הגיוני של הקלאסטר הזה.

K-Means:  
בסעיף הקודם דנו בקבלת ההחלטה לגבי ערך הפרמטר K. מעבר לפרמטר זה, הגדרנו את הפרמטר init, אשר שולט על האופן שבו מאותחלים הצנטרואידים. ברירת המחדל היא שהאלגוריתם מאתחל את הצנטרואידים באופן שרירותי, וכך עשויים להתקבל קלאסטרים סמוכים מאוד אחד לשני (כלומר, כאלה שהיו עשויים להתאים לקלאסטר אחד), ובנוסף, ההתכנסות של המודל עשוייה לקחת זמן רב. אלגוריתם בחירת צנטרואידים אחר שחבילת sklearn מאפשר היא k-means++, אשר מבצעת אופטימזציה של הערכים ההתחלתיים של הצנטרואידים.

LogisticRegression:  
פרמטר מרכזי באלגוריתם לפיתוח מודל Logisitic Regression הוא הפרמטר C, אשר נועד "להעניש" את האלגוריתם כאשר המשקלים הפנימיים שלו הופכים גדולים מדי. גידול זה מאפיין לרוב overfitting ולכן נדרשת מידה של זהירות בהגדלת הערכים. ניסינו את ערכי C הבאים: 0.01, 0.05, 0.25, 0.5 ו-1. את ערכי ה-accuracy הטובים ביותר ביחס ל-test, קיבלנו עבור הערך C=0.25.

**מסקנות:**

השימוש במודלים של חבילת sklearn אינטואטיביים מאוד, וקלים להבנה. בכדי להפיק את המירב ממודלים אלו יש צורך כמובן בידע תיאורטי קודם. אם ניתן לשפר משהו בחבילה זו, זהו אולי האתר של החבילה שמציג את התיעוד. התיעוד מעט יבש, ומניח שהקורא בקי בטרמינולוגיה הפנימית של sklearn. מצאנו את עצמינו פעמים רבות מחפשים להבין מה המשמעות של פרמטר כזה או אחר.

השימוש בDask מתיימר להיות פשוט כמו Pandas, אך פעמים רבות מצאנו את עצמנו לא מצליחים להגיע למידע עצמו, או משרשרים רצץ של פעולות רבות בכדי לקבל בסופו של דבר פריט מידע יחסית טריוויאלי. לפי הכתוב באתר של החבילה, מדבור בחבילה חזקה, ושאומצה בחום על ידי קהילת הפיתוח ובתעשייה, אך אנו התקשינו מאוד בשימוש בה, ולמעשה בכל הזדמנות שהייתה לנו, העדפנו לעבוד עם pandas על חלקים מתוך הdataset השלם.

העבודה עם Tkinter מאוד אינטואיטיבית גם כן . למרות שמצאנו כבר דברים אשר מוגבלים מבחינת עיצובית בחבילה – היא מכילה פונקציות וגישות לערכים השונים (כמו הקלט של המשתמש לדוגמא) בצורה נוחה מאוד.

לסיכום, למדנו המון בעבודה הזו. החל מהחבילות אליהן נחשפנו ועד לבניית המודלים השונים.