מבוא למדעי המחשב 67101 - סמסטר א' 2021/2022

תרגיל 6 - Moogle חיפוש ודירוג דפי אינטרנט

להגשה בתאריך **24/11/2021** בשעה 22:00

הקדמה

בתרגיל זה נכתוב מעין מנוע חיפוש שיאפשר חיפוש באוסף דפי אינטרנט. נעשה זאת על ידי הורדת דפים מהרשת ומימוש אלגוריתם Page Rank המפורסם של גוגל. לאורך כל התרגיל נעבוד עם ספריות לחיפוש וניתוח תוכן בדפי אינטרנט, וכן נממש שיטות לדירוג אתרי אינטרנט לפי חשיבות, וכן חיפוש בהם.

דגשים לתרגיל

- אין להשתמש בספריות **numpy, scipy, pandas** וכל ספרייה אחרת, ללא אישור מפורש בפורום.
- התוכנית תופעל דרך הקובץ **moogle.py**, כפי שיפורט בהמשך. מומלץ לפרק את התוכנית לקבצים נוספים ולהגישם גם כן. את החלוקה לקבצים ולפונקציות עליכם לבצע על פי שיקול דעתכם, לפי העקרונות שנלמדו בקורס. הדגש צריך להיות על קוד מודולרי (ללא כפל קוד), ברור וקריא.
- כל הקבצים יוגשו בתוך קובץ **ex6.zip**. ניתן להניח שכל הקבצים המוגשים בקובץ זה יימצאו בתיקיה בעת הקריאה לסקריפט. שימו לב לא להגיש קבצים מיותרים (בפרט את קבצי הסביבה הווירטואלית או קבצי קלט/פלט)
- סגנון: הקפידו על תיעוד נאות ובחרו שמות משתנים משמעותיים. הקפידו להשתמש בקבועים (שמות משתנים באותיות גדולות) על פי הצורך.
- רקע על אלגוריתם Page Rank, ורקע אודות דפי HTML וחבילת BeautifulSoup נמצא בשקפי תרגול 6, וכן בקובץ עזר במודל.
- מומלץ להריץ את הקוד בסביבה וירטואלית, המכילה את הספריות הנדרשות בתרגיל (**requests, bs4**). ניתן להתקין את הספריות בעזרת `pip install requests bs4`
- מותר (אך לא חובה) להשתמש בספריות **Collections** ו-**argparse**

תיאור התרגיל

בתרגיל זה אתם מתבקשים לבנות מנוע חיפוש בסיסי, הנקרא **Moogle**, בו ניתן לחפש ביטויים שונים בתוך אוסף דפים מתוך הויקי של הארי פוטר (ניתן לעיין בדפי המקור פה). היות והויקי המלא הוא מאוד גדול יצרנו עבורכם קבוצה קטנה יותר של דפים אליהם ניתן לגשת בכתובת

https://www.cs.huji.ac.il/~intro2cs1/ex6/wiki/some_page_name

תיאור התרגיל כולל מספר חלקים:

- חלק (א) - הורדת הדפים מהאינטרנט וגילוי הקישורים בינם
- חלק (ב) - דירוג אוסף הדפים לפי חשיבות
- חלק (ג) - יצירת מיפוי ממילים לדפים בהם הן מופיעות
- חלק (ד) - ביצוע חיפושים של מילים בדפים, על בסיס החלקים הקודמים

חלק (א) - גישה אל הדפים דרך הרשת וגילוי הקישורים בינם

בחלק הראשון של התרגיל, ניצור מילון המפרט את הקישורים בין הדפים אותם אנו סורקים.

כדי לבנות את מילון הקישורים, נקרא לתוכנית **moogle.py** משורת הפקודה באופן הבא:

```
python3 moogle.py crawl <BASE_URL> <INDEX_FILE> <OUT_FILE>
```

כאשר:

- **<BASE_URL>** היא כתובת אתר האינטרנט בו נמצאים הדפים
- **<INDEX_FILE>** הינו שם של קובץ טקסט אשר מכיל רשימת דפים בתוך האתר הנתון **Harry_Potter.html**, **Albus_Dumbledore.html** ועוד.
- **<OUT_FILE>** הינו שמו של קובץ הפלט בו תשמרו את טבלת הקישורים.

לדוגמא הפקודה:

```
python3 moogle.py crawl
https://www.cs.huji.ac.il/~intro2cs1/ex6/wiki/ small_index.txt
out.pickle
```

תאסוף קישורים מהאתר של מדעי המחשב (הדפים שמופיעים בקובץ `small_index.txt`) ותשמור את התוצאה בקובץ `out.pickle`.

לנוחיותכם, סיפקנו לכם קובץ בשם `small_index.txt` המכיל רשימה של 12 דפי אינטרנט שונים בוויקי שלנו (הנתונים ככתובות יחסיות). הכתובת היחסית באתר מדעי המחשב בה נמצא דף הויקי של הארי פוטר היא:

```
/~intro2cs1/ex6/wiki/Harry_Potter
```

כדי לחבר יחד כתובת של אתר (BASE_URL) וכתובת יחסית יש לכתוב:

```
import urllib.parse
full_url = urllib.parse.urljoin(base_url, relative_url)
```

מילון הקישורים

מילון הקישורים שלנו יהיה בעל המבנה הבא: ניצור מבנה נתונים traffic_dict שבעזרתו נוכל לגלות כמה קישורים קיימים מדף אינטרנט ששמו page_name לכל אחד מדפי האינטרנט האחרים בויקי.

נוכל לעשות זאת באמצעות שימוש במילונים של פייתון. הטיפוס של traffic_dict יהיה:

traffic_dict: Dict[str, Dict[str, int]]

כלומר מילון הממפה מחרוזת למילון.

נרצה ש:

[traffic_dict[page_name][linked_page_name יתן לנו את מספר הפעמים שהדף האינטרנט ששמו page_name מכיל קישורים לדף ששמו linked_page_name.

למשל:

["traffic_dict["Harry_Potter.html"]["Tom_Riddle.html יהיה מספר הקישורים שקיימים בדף אינטרנט ששמו "Harry_Potter.html" לדף אינטרנט ששמו "Tom_Riddle.html".

הערות:

- קישור מדף A לדף B, הוא לינק המצויין על-ידי תגית הקישור <a>, בקוד ה-HTML של דף A. בתוך התגית מופיע הערך href שאליו משוייכת הכתובת של דף B. לדוגמא:

```
<a href="http://www.google.com">google</a>
```

  הוא קישור למנוע החיפוש המתחרה.
- קישורים בתוך אותו אתר יכולים להיות יחסיים. למשל הקישור
  `<a href="Harry_Potter.html">harry</a>` מצביע לדף
  `Harry_Potter.html` בתוך אותו אתר (BASE_URL).
- דף יכול להצביע לעצמו.
- יש לקחת בחשבון כפילויות. דף כלשהו יכול לכלול יותר מקישור בודד לכל יעד.

- אנחנו נסרוק אך ורק דפים שהינם מתוך קבוצת דפי הויקי של הארי פוטר ששמותיהם מופיעים בקובץ **<INDEX_FILE>**. יש להתייחס רק לקישורים יחסיים באותו אתר ומאלה יש להחשיב רק כאלו שמופיעים בקובץ האינדקס שניתן בשורת הפקודה. יש להתעלם מקישורים אחרים בדף. הם לא יספרו במילון.
- ניתן להניח שלכל דף יש לפחות קישור יוצא אחד (רלוונטי לדירוג האתרים בהמשך התרגיל).

הנחיות נוספות:

- כדי לקבל את קוד ה-HTML של דף אינטרנט מסוים, עליכם להגיש בקשת HTTP לשרת האינטרנט המכיל את הדף. לשם כך תוכלו להשתמש בספריה **requests** באופן הבא:

```
response = requests.get(url_path)
html = response.text
```

כאשר url_path הוא משתנה המכיל את כתובת אינטרנט המלאה לדף האינטרנט שאנו מעוניינים לקבל את קוד ה-HTML שלו. לתוך המשתנה html יכנס קוד ה-HTML עצמו, והוא יהיה משתנה מטיפוס מחרוזת (str).

- נזכיר כי אנו נתמקד בתגיות <a> שהינן מקוננות בתגיות הפסקה בלבד, בקוד ה-HTML של דף האינטרנט. תוכלו להשתמש בספריה bs4 כדי לגשת לכל הלינקים המופיעים בתוך פסקאות כך: אם למשל המשתנה html מכיל את קוד HTML של דף אינטרנט מסויים, תוכלו לעבור על כל הפסקאות והלינקים שבתוך פסקאות בעזרת הפונקציה find_all(), ולקבל את ערך התכונה href של כל לינק, כך:

```
soup = bs4.BeautifulSoup(html, 'html.parser')
for p in soup.find_all("p"):
      for link in p.find_all("a"):
            target = link.get("href")
```

בדוגמה זו, בכל איטרציה של הלולאה הפנימית, המשתנה target יקבל את ערך התכונה href של הלינק כלומר את הכתובת של דף היעד של הקישור.

<u>שמירת המילון</u>

על התוכנית לשמור את מילון הקישורים שיצרתם. אנחנו נשמור את הקובץ בפורמט של קובץ **pickle**, תחת השם המתקבל בפרמטר **<OUT_FILE>** משורת הפקודה. שמירה בפורמט זה (ולא בקובץ טקסט) מאפשרת לשמור את המילון בצורה קומפקטית וקלה. תוכלו לקרוא עוד על פורמט **pickle** פה.

כדי לשמור את המילון בקובץ **pickle**, יש צורך לפתוח אותו במצב כתיבה בינארית, ולאחר מכן לקרוא לפונקציה pickle.dump() כך:

```
with open(filename, 'wb') as f:
    pickle.dump(d, f)
```

בדוגמה זו המשתנה filename מכיל את שם קובץ הפלט שאליו יישמר המילון, והמשתנה d מכיל את המילון שברצונכם לשמור.

חלק (ב) - דירוג דפי אינטרנט לפי חשיבות

בחלק השני, נעשה שימוש במילון הקישורים כדי לדרג את דפי האינטרנט שלנו. הדירוג של האתרים ייעשה על ידי גרסא פשוטה של אלגוריתם Page Rank של גוגל, שראינו בתרגול, אשר משמש לדירוג אתרים לפי מספר "הצבעות" יחסי, כלומר: ככל שדף אינטרנט X הינו דף שמצביעים עליו יותר אתרים אחרים, כך הדירוג של X יהיה גבוה יותר ביחס לשאר הדפים.

הדירוג ישמר במילון של פייתון, כאשר כל מפתח הוא שם של דף אינטרנט, כפי שהופיע בקובץ <INDEX_FILE> שהתקבל כפרמטר בשורת הפקודה בחלק א'. הערך של כל מפתח יכיל את דירוג ה-Page Rank של אותו דף. כדי ליצור את מילון הדירוגים, נקרא לתוכנית **moogle.py** משורת הפקודה באופן הבא:

```
python3 moogle.py page_rank <ITERATIONS> <DICT_FILE> <OUT_FILE>
```

כאשר:

- <ITERATIONS> הוא מספר שלם אי-שלילי המתאר את מספר האיטרציות להרצת אלגוריתם ה- Page Rank, כפי שיפורט בהמשך.
- <DICT_FILE> הוא שמו של קובץ pickle המכיל את מילון הקישורים באותו פורמט שיצרתם בחלק א'
- <OUT_FILE> הוא שמו של קובץ הפלט שבו ישמר מילון הדירוגים, גם כן בפורמט pickle.

כעת נרחיב על אלגוריתם Page Rank.

שלב 1 - יצירת דירוג לדפי האינטרנט

כעת, אנחנו נתחיל ביצירה של מדרג חשיבות בין אתרי האינטרנט השונים, באוסף הדפים שהורדנו. עבור N דפי אינטרנט, אנחנו ניצור מילון **r** בעל N רשומות, ונעדכן בכל איטרציה את ערכיו של המילון **r**, שהינם מטיפוס float. הטיפוס של המילון r יהיה:

r: Dict[str, float]

מספר האיטרציות שבהם נעדכן את ערכי **r** הינו הערך <ITERATIONS>, המתקבל כפרמטר בשורת הפקודה בעת הפעלת התוכנית.

אפן החישוב של מילון הדירוגים **r** ייעשה בשלבים, לפי התיאור הבא:

1. תחילה, כל דף אינטרנט יקבל דירוג שווה בעל הערך 1.
2. בכל איטרציה (סיבוב) נעדכן את המילון **r** באופן הבא:

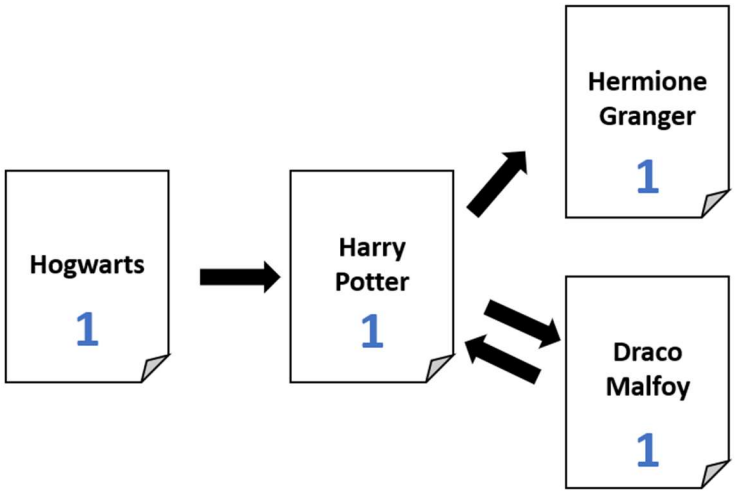
- הדירוג של כל דף יתחיל כ-0.
- כל דף יחלק את הדירוג שקיבל בסיבוב הקודם (בסיבוב הראשון זה 1, ובהמשך ערך אי-שלילי אחר) בין כלל הדפים שעליהם הוא מצביע.
- כלומר, עבור דף **i** שמצביע לדף **j** כלשהו, הדף **j** יקבל מהדף **i** את הערך הבא:

$$new\_r[j] += r[i] \times \frac{< num\ links\ i \rightarrow j >}{< total\ num\ links\ from\ i >}$$

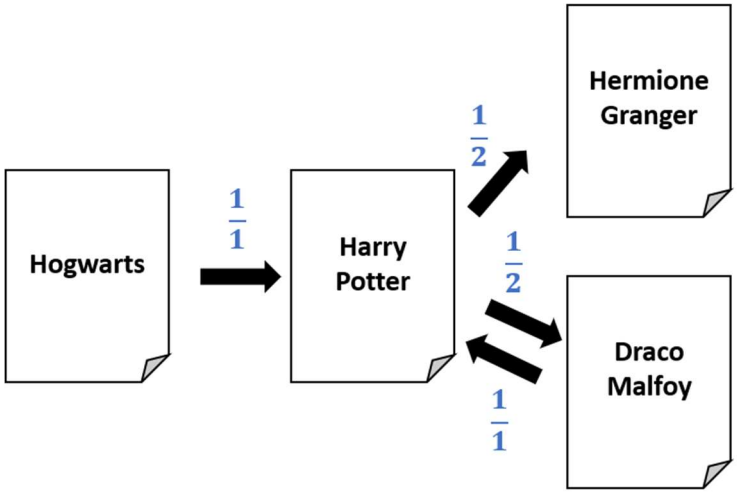
שימו לב שהערכים בשבר הנ"ל ניתנים לחישוב בקלות ממילון הקישורים בין הדפים.

- בסוף האיטרציה כל דף מקבל למעשה את סכום הערכים שתרמו לו הדפים שמצביעים עליו (מהדירוג שלהם בסיבוב הקודם). שימו לב שהערך של כל דף יכול לקטון \ לגדול \ לא להשתנות בכל סיבוב, כתלות בערכים של הדפים האחרים, אך הינו תמיד אי-שלילי.

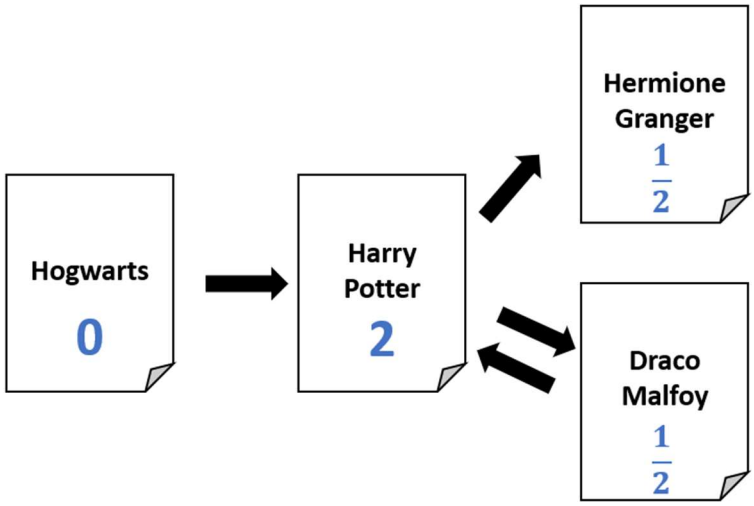
למשל, הסקיצה הבאה מראה דף במרכז, ששמו **Harry Potter** אשר כולל שתי הצבעות יוצאות: הצבעה אחת לדף **Hermione Granger**, והצבעה אחת לדף **Draco Malfoy**, וכן שתי הצבעות נכנסות מהדפים **Hogwarts** ו-**Draco Malfoy**.



בתחילת האיטרציה הראשונה לדף **Harry Potter** יהיה ערך **1**, והוא יעניק $(\mathbf{1} * \frac{\mathbf{1}}{\mathbf{2}})$ לדף **Hermione Granger**, ויעניק $(\mathbf{1} * \frac{\mathbf{1}}{\mathbf{2}})$ לדף **Draco Malfoy**. כמו כן, הדף **Harry Potter** הינו בעצמו דף שיקבל דירוגים מהדפים אשר מצביעים אליו. בדוגמה הזו, הדף **Harry Potter** יקבל $(\mathbf{1} * \frac{\mathbf{1}}{\mathbf{1}})$ מהדף **Hogwarts**, ו- $(\mathbf{1} * \frac{\mathbf{1}}{\mathbf{1}})$ מהדף **Draco Malfoy** . הציור הבא ממחיש את הענקת הדירוגים של הדף **Harry Potter** לדפים אחרים, ושל הדפים האחרים אליו:



בסיום האיטרציה הראשונה, מצב הערכים של כל דף בדוגמה הנ"ל יהיה כדלקמן:



לאחר ביצעו העדכון כמספר האיטרציות שמתקבל בפרטמר `<ITERATIONS>`, אנחנו נקבל לבסוף מילון **r**, אשר אמור לקיים:

- מכיל ערכים אי שליליים.
- מכיל ערכים נסכמים לערך שהינו נומרית קרוב מאוד ל-N, מספר הרשומות במילון הקישורים (תחשבו למה!)

מילון זה ייצג לנו את הדירוגים הפנימיים של N האתרים השונים - ככל שאתר יותר חשוב, ומצביעים אליו יותר אתרים אחרים, כך הוא צפוי לקבל ערך גבוהה יחסית, ואילו אתרים אשר הינם פחות חשובים, ומצביעים אליהם פחות אתרים אחרים - ידורגו עם ערך נמוך יותר (קרוב ל-0).

שימו לב, שעבור מספר גבוה של איטרציות, אנחנו צפויים לראות התכנסות של ערכי המילון **r** למילון עם ערכים (כמעט) קבועים, כלומר המילון **r** יכלול ערכים שלא ישתנו אחרי איטרצית עדכון.

הנחיות נוספות:

- על התוכנית שלכם ליצר מילון דירוגים בהינתן מילון הקישורים, השמור בפורמט **pickle** שנוצר בחלק א'. שם קובץ ה-pickle המכיל את מילון הקישורים מתקבל בפרמטר **<DICT_FILE>** הניתן בשורת הפקודה בהרצת התוכנית.
- כדי לטעון את מילון הקישורים מתוך קובץ **pickle** למילון של פייתון, עליכם לפתוח את הקובץ במצב קריאה בינארית, ולטעון את תוכן הקובץ בעזרת הפונקציה pickle.load, כפי שניתן לראות בקוד הבא:

```
with open(filename, "rb") as f:
    d = pickle.load(f)
```

בדוגמה זו המשתנה filename מכיל את שם קובץ הקלט שממנו נקרא את המילון, ולמשתנה d יכנס המילון שברצונכם לקרוא.

<u>שמירת מילון הדירוגים</u>

לבסוף, על התוכנית לשמור את תוצאות מילון הדירוגים **r**, בסוף ריצת האלגוריתם. יש לשמור את המילון לקובץ בפורמט **pickle** גם כן, בדומה לחלק א'. שם קובץ הפלט ניתן בפרמטר **<OUT_FILE>** משורת הפקודה.

חלק (ג) - יצירת מיפוי ממילים לדפים

כדי לחפש במנוע החיפוש שלנו, נרצה לדעת עבור מילים שיופיעו בשאילתת חיפוש אילו דפי אינטרנט רלוונטיים לחיפוש. ניצור מבנה נתונים word_dict (שמבנהו דומה למבנה מילון הקישורים שתיארנו בחלק א') שבעזרתו נוכל לגלות כמה פעמים מילה word הופיעה בדף ששמו page_name.

נוכל לעשות זאת באמצעות שימוש במילונים. הטיפוס של word_dict יהיה:

word_dict: Dict[str, Dict[str, int]]

כלומר מילון הממפה מחרוזת, למילון.

נרצה ש:

[word_dict[word][page_name יתן לנו את מספר הפעמים שהמילה word מופיעה בדף ששמו page_name.

למשל:

["word_dict["Harry"]["Ronald_Weasley.html יהיה מספר המופעים של המילה "Harry" בדף ששמו "Ronald_Weasley.html" ברשימת הדפים שלנו.

שימו לב: היות ורוב המילים לא מופיעות בהרבה דפים, אין צורך להכניס רשומות 0 למילון. כלומר: אם אין רשומה במילון עבור מילה מסויימת (או אין רשומה לדף מסוים במילון הפנימי) פירוש הדבר שהמילה אינה מופיעה כלל בדפים או אינה מופיעה כלל בדף הספיציפי.

כדי לבנות את מילון המילים, נקרא לתוכנית **moogle.py** משורת הפקודה באופן הבא:

```
python3 moogle.py words_dict <BASE_URL> <INDEX_FILE> <OUT_FILE>
```

כאשר:

- **<BASE_URL>** היא כתובת האתר בו נמצאים הדפים שיש לגשת אליהם
- **<INDEX_FILE>** הינו קובץ טקסט אשר מכיל רשימת שמות של דפי אינטרנט שונים (בדומה לחלק א')
- **<OUT_FILE>** הינו מסלול לקובץ פלט בו תשמרו את מילון המילים, בפורמט **pickle**.

נייצר את מילון המילים על-ידי מעבר על כל פסקאות הטקסט ברשימת הדפים שבקובץ **<INDEX_FILE>**. בשלב השני נשמור את המילון לקובץ **pickle**, בדומה לחלקים הקודמים.

<u>יצירת המילון</u>

עליכם לגשת, בעזרת בקשת HTTP, לכל דף ברשימת דפי האינטרנט המתקבלים בפרמטר **<INDEX_FILE>** משורת הפקודה, ולעבור על כל הפסקאות שמכיל כל דף אינטרנט. מכל פסקה נוציא את כל רשימת המילים המופיעות בה, ונעדכן בהתאם את מילון המילים לפי ההנחיות הבאות:

- עליכם לגשת לדפים ששמותיהם מופיעים בקובץ **<INDEX_FILE>** דרך בקשות HTTP בעזרת הספריה **requests** (בדומה לחלק א').
- מכל אחד מהדפים, עליכם לאסוף את כל המילים המופיעים בתוכן של תגיות הפסקה בלבד. השתמשו בספריה **bs4** בפונציה bs4.findall() כדי למצוא את כל תגיות בכל מסמך. על כל אוביקט p של תגית הקיימת במסמך, ניתן לגשת לערך p.text שמכיל את התוכן של התגית. למשל, בקטע הקוד הבא:

```
for p in soup.findall("p"):
    content = p.text
```

הלולאה עוברת על כל תגית פסקה הקיימת במסמך, והמשתנה content יכיל את התוכן של אותה פסקה.

- המילים בטקסט מופרדות ע"י whitespaces. יש לנקות רווחים טאבים וירידות שורה מתחילת וסוף כל מילה. (העזרו ב-str.split לצורך ההפרדה למילים).
- אין צורך לסנן את המילים בתוך הפסקאות. ייתכן שיופיעו בהן תגיות HTML וכל מיני דברים מוזרים אחרים. זה לא יפריע לחיפוש מילים תקינות.

- מילה שנכתבת ב-2 צורות שונות עם אותיות גדולות וקטנות תחשב ל-2 מילים נפרדות. למשל המילים "Harry", "HARRY" ו-"harry" יחשבו ל-3 מילים שונות במילון.
- כמו כן, מילים שאליהן צמודים סימני פיסוק יחשבו כמילים שונות מאלו שללא סימני פיסוק. למשל המילים "harry" ו-"harry," יחשבו כמילים שונות.

שמירת מילון המילים

לבסוף, על התוכנית לשמור את מילון המילים שיצרתם לקובץ פלט. אנחנו נשמור את המילון לקובץ בפורמט **pickle** גם כן, בדומה לחלקים הקודמים. שם קובץ הפלט ניתן בפרמטר **<OUT_FILE>** משורת הפקודה.

חלק (ד) - ביצוע חיפוש ודירוג התוצאות

בחלק זה, אנחנו נתבסס על קובץ מילון הדירוגים שיצרתם בסוף חלק ב', וכן על מילון המילים שיצרתם בחלק ג', ונשתמש בהם כדי ליצור את מנוע החיפוש שלנו - Moogle.

שאילתת חיפוש תופעל משורת הפקודה. יתבצע סינון ודירוג של התוצאות, ולבסוף תתבצע הדפסה של תוצאות החיפוש.

הרצת הקובץ משורת הפקודות

את מנוע החיפוש נריץ משורת הפקודה עם הפרמטרים הבאים:

```
python moogle.py search <QUERY> <RANKING_DICT_FILE>
<WORDS_DICT_FILE> <MAX_RESULTS>
```

כאשר:

- **<QUERY>** - היא שאילתת החיפוש, המתקבלת בפורמט שיפורט בהמשך.
- **<RANKING_DICT_FILE>** - מסלול לקובץ מילון הדירוגים, כפי שיצרתם ושמרתם בחלק (ב).
- **<WORDS_DICT_FILE>** - מסלול לקובץ מילון המילים, כפי שיצרתם ושמרתם בחלק (ג).
- **<MAX_RESULTS>** - שהינו מספר טבעי המייצג לנו את מספר התוצאות המקסימאלי עבור החיפוש, כפי שיפורט בהמשך.

שאילתת החיפוש, המתקבלת בפרמטר **<QUERY>** הינה מחרוזת של מספר מילים בודדות, מופרדות בתו רווח בודד. על מנוע החיפוש שלכם להציג תוצאות חיפוש אשר יכילו את **כל** המילים של שאילתת החיפוש.

סינון התוצאות לפי ציון משוקלל, והדפסת התוצאות עבור החיפוש

כעת, נשתמש בשלושת הארגומנטים הנוספים שאיתם קראנו לתוכנית משורת הפקודות, **<WORDS_DICT_FILE>**, **<RANKING_DICT_FILE>** ו-**<MAX_RESULTS>** על מנת לדרג את התוצאות למשתמש.

תהליך זה יתבצע לאחר בחירת **<MAX_RESULTS>** הדפים הראשונים מתוך ויקי הארי פוטר שמכילים את כל מילות החיפוש, ממויינים לפי הציון הניתן להם במילון הדירוגים שהתקבל בפרמטר **<RANKING_DICT_FILE>**, אשר מקיימים את תנאי החיפוש שהוגדרו בשלב הקודם. במידה ויש פחות מ- **<MAX_RESULTS>** תוצאות, אז יש לבחור רק את תוצאות אלה.

לאחר בחירת התוצאות, יש להדפיס את שמות הדפים שבהם נמצאו התוצאות למשתמש, לפי דירוג משוקלל, אשר מבוסס גם על ערך החשיבות של כל דף (במילון הדירוגים ב-**<RANKING_DICT_FILE>**) וגם על מספר ההופעות של הביטוי עצמו באותו דף (שאפשר לקבל ממילון המילים ב-**<WORDS_DICT_FILE>**).

בדירוג שלנו, הציון המשוקלל של כל דף בנוי מהמכפלה של ערך הדף במילון הדירוגים, כפול מספר ההופעות של מילות החיפוש בדף. למשל - אם נחפש את המילה **Scar** בדף **Harry_Potter**.html וכן:

- ✓ ערך הדף **Harry_Potter**.html במילון הדירוגים הינו Y
- ✓ מספר ההופעות של **Scar** בדף זה הינו Z

אזי הציון המשוקלל של הדף **Harry_Potter**.html, ספציפית עבור החיפוש של **Scar**, הינו: Y*Z. עבור שאילתות חיפוש המכילות מספר מילים, מספר ההופעות הכולל יקבע כמספר ההופעות **המינימלי** מבין כל אחת מהמילים הבודדות.

פורמט הפלט הינו:

<page1> <score1>
<page2> <score2>
...

כאשר בכל שורה יש את שם דף הויקי (Harry_Potter.html, Albus_Dumbledore.html וכו') ע"פ הציון המשוקלל, וכן הציון המשוקלל עצמו, מופרדים בתו רווח בודד, וממוינים מהערך הגבוה לנמוך.

הנחיות נוספות:

- שימו לב כי ייתכן שמילון הדירוגים ומילון המילים **נוצרו על-ידי קבצי אינדקס שונים!** פירוש הדבר הוא שייתכן כי בחיפוש של מילה מסויימת במילון המילים יופיע שם של דף אינטרנט שאין לו דירוג במילון הדירוגים, וגם להפך – דף אינטרנט ששמו מופיע במילון הדירוגים, אך אינו מכיל אף מילה במילון המילים, ולכן שמו לא מופיע שם.
- יש להציג בתוצאות החיפוש רק דפי אינטרנט ששמם מופיע במילון הדירוגים **וגם** במילון המילים, עבור מילות החיפוש שהוגדרו בשאילתת החיפוש.
- אם אחת ממילות החיפוש אינה מופיעה במילון המילים, יש להתעלם ממנה, ולהתייחס לשאילתת החיפוש כאילו היא לא מכילה את אותה מילה.

שאילתות חיפוש לדוגמא

עליכם לבצע חיפוש עבור השאילתות הבאות, ולהוציא את התוצאות לקובץ ***results.txt***:

1. scar
2. Crookshanks
3. Horcrux
4. Pensieve McGonagall
5. broom wand cape

הקובץ ***results.txt*** צריך להיות בעל הפורמט הבא:

```
<OUTPUT OF SEARCH QUERY 1>
**********
<OUTPUT OF SEARCH QUERY 2>
**********
<OUTPUT OF SEARCH QUERY 3>
**********
<OUTPUT OF SEARCH QUERY 4>
**********
<OUTPUT OF SEARCH QUERY 5>
**********
```

יש להריץ את אלגוריתם ה-Page Rank ל-100 איטרציות, ולהציג (עד) ארבעת התוצאות עם הציון המשוקלל הגבוה ביותר.
כלומר הפלט של התוכנית עבור כל שאילתת חיפוש, ולאחריה שורה המכילה עשר כוכביות, ללא שורות רווח לפני ואחרי פלט של כל שאילתה ושורת כוכביות.

מספר נקודות העשרה

אלגוריתם Page Rank ניתן לכתיבה בצורה פשוטה יותר בעזרת כלים של אלגברה לינארית.

אפשר להתייחס למילון הדירוגים r בתור וקטור, ולמילון הקישורים כמטריצה. כאשר כל איטרציה באלגוריתם שקולה למעשה למכפלה של הווקטור במטריצה (לאחר ששורותיה מנורמלות). למעשה, אפשר גם להראות שהאלגוריתם שקול לגולש שמבצע "הילוך מיקרי" על דפי האינטרנט ולוחץ על קישורים באקראי. הדירוג שהאלגוריתם נותן לכל דף נמצא ביחס ישר למספר הפעמים שהגולש האקראי יבקר בכל דף! ראו עוד כאן.

תהליך העדכון של Page Rank מתכנס על פי רוב וניתן להראות זאת בעזרת כלים של אלגברה לינארית (על פי משפט פרון-פורביניוס)

בשלב זה בנינו את הגרסא הפשוטה של אלגוריתם Page Rank המפורסם ושל מנוע החיפוש, שהוצג לעולם בשנת 1998 על-ידי המוגלגים לארי פייג' וסרגיי ברין, בראשיתה של חברת גוגל.

הוראות הגשה

עליכם להגיש את הקובץ **ex6.zip** (בלבד) בקישור ההגשה של תרגיל 6 דרך אתר הקורס על ידי לחיצה על "Upload file". אנו ממליצים להתחיל לעבוד על התרגיל בשלב מוקדם.

ex6.zip צריך לכלול לפחות את ~~הקובץ~~ הקבצים **moogle.py** ו-**results.txt**, אך יכול גם להכיל קבצי פייתון נוספים במידה והתוכנית שלכם פוצלה על כמה קבצים. שימו לב לא לכלול קבצי קלט/פלט וקבצים של הסביבה הווירטואלית.

כמו כן, אין להגיש את הקובץ small_index.txt.

הנחיות כלליות בנוגע להגשה

- הנכם רשאים להגיש תרגילים דרך מערכת ההגשות באתר הקורס מספר רב של פעמים. ההגשה האחרונה בלבד היא זו שקובעת ושתיבדק.
- לאחר הגשת התרגיל, ניתן ומומלץ להוריד את התרגיל המוגש ולוודא כי הקבצים המוגשים הם אלו שהתכוונתם להגיש וכי הקוד עובד על פי ציפיותיכם.
- באחריותכם לוודא כי - PDF הבדיקות נראה כמו שצריך.
- קראו היטב את קובץ נהלי הקורס לגבי הנחיות נוספות להגשת התרגילים.
- שימו לב - יש להגיש את התרגילים בזמן!

בהצלחה!