ניהול נתונים באינטרנט תרגיל תכנותי 1

מגישים: גיא גבריאלי ת.ז 323832204 , רביע אזרייק ת.ז 211710124

חלק ראשון - תיאור הקוד שבונה את האונטולוגיות:

בחלק הראשון אנחנו בונים את האונטולוגיות, זה קורה באופן הבא:

תחילה אנחנו מאתחלים תור עם כל הurlים של מדינות מהטבלה שקיבלנו, ולאחר מכן אנחנו עושים crawling על מנת לחלץ את הדאטה הרלוונטי,

הפונקציות שכתבנו:

extract_name(url)
פונקציה זו מקבלת url ולוקחת את החלק האחרון שלו, לדוגמה https:/wiki/israel, הפונקציה תחזיר את israel

create_initial_urls_queue()
פונקציה זו מבצעת שאילתת xpath על עמוד ה HTML שמצורף במטלה (List of countries by population) על מנת לחלץ את כל הלינקים לעמודי הויקיפדיה של המדינות מהטבלה, ומכניסה אותם לתור לפי הסדר כטאפלים של הurlים בצירוף הסטרינג country לתור.

insert_to_graph(object1,relation,object2)
הארגומנטים של פונקציה זו הם: object1 שהוא רשימה של ישויות(בן אדם/מדינה) שרוצים להכניס לגרף וrelation שזה הקשר בין הישויות לדוגמה president of ובנוסף מקבלת ישות(object 2) ומכניסה לגרף את כל הקומבינציות של var,relation,object2 כאשר var הוא איבר ברשימה object1 לאחר שעשינו לו מספר שינויים קטנים(כמו להחליף רווחים בקו תחתון וכו)

insert_tuples_to_queue(label , links)
פונקציה זו מקבלת label(כאשר label הוא ישות כלשהו כמו Prime minister , countryוכו) ורשימת לינקים ומכניסה לתור את הטפלים label,link כאשר link הוא איבר בlinks, משתמשים בזה כשרוצים להכניס הרבה ישויות לתור שיש להם את אותו סוג ישות, למשל אם יש הרבה לינקים לנשיאים אז מכניסים את כולם לתור עם התגית של נשיא.

country_label_handler(url,label)
פונקציה זו מקבלת url של מדינה וlabel ומבצעת שאילתת xpath על מנת לחלץ את המידע הרלוונטי ולהכניס אותו לתור/גרף
לדוגמה פונקציה זו יכולה לקבל את ה url של עמוד הויקיפדיה של סין, וlabel = prime minister, והיא תבצע את השאילתה הדרושה על מנת לחלץ את ראש הממשלה של סין, ותכניס את התוצאה לגרף האונטולוגיות, ותכניס את הurl של ראש ממשלת סין לתור כדי שנוכל להמשיך לחלץ מידע גם מדף הויקיפדיה שלו

person_label_handler(url , label)

הלוגיקה של פונקציה זו דומה מאוד לזו של הפונקציה country_label_handler רק שהיא מטפלת בurlים של נשיאים/ראשי ממשלה וכו במקום urlים של מדינות

web_crawler()
פונקציה זו היא זו שאחראית לבצע את הסיור בדפי האינטרנט, כל עוד התור שלנו לא ריק(=יש urlים שצריך לבקר בהם) פונקציה זו עושה pop לtuple מהתור (url,type) כאשר type = country/president/prime_minister וקוראת לפונקציה country_label_handler או person_label_handler בהתאם לtype ופונקציות אלה ממשיכות להכניס לינקים לתור עד שמסיימים את התהליך של ה crawling והתור נהיה ריק עוצרים

create_ontology()
פונקציה זו קוראת לשתי פונקציות - create_initial_urls_queue על מנת לאתחל את התור שיכיל את הטאפלים של כל הurlים של המדינות(בצירוף type = country)
ולאחר מכן קוראת לweb_crawler שאחראית לסייר על עמודי הויקיפדיה ולבנות את גרף האונטולוגיות שלנו

חלק שני - תיאור של השאלה שהוספנו ודוגמאות:

השאלה שהוספנו היא: How many prime ministers were born in COUNTRY השאילתה תחזיר את כמות הראשי ממשלה שנולדו במדינה מסוימת, דוגמאות:

```
nova 63% python3 geo_qa.py question "How many prime ministers were born in Algeria?"
1
nova 64% python3 geo_qa.py question "How many prime ministers were born in France?"
4
nova 65% python3 geo_qa.py question "How many prime ministers were born in Israel?"
1
nova 66% python3 geo_qa.py question "How many prime ministers were born in Brazil?"
0
```

חלק שלישי - תיאור מקרי קצה שהתמודדנו איתם:

1. לפעמים ערכי החזרה של שאילתות ה XPATH שקיבלנו היו דברים שאפשר בקלות להבחין שהם לא רלוונטים ושהם במקרה תאמו לשאילתה והוחזרו, וברוב המקרים האלה הייתה חזרה (למשל הם תמיד התחילו ב) או] או / או geohack או #cite…) אז הייתה לנו רשימת blacklist שכל משהן שמוחזר שמכיל את הערכים של הרשימה הזאת אז הוא ערך לא רלוונטי ולכן זרקנו אותו. למשל עשינו את הבדיקות האלה על ערך החזרה ואם התנאי התקיים אז זורקים אותם
```
("object1[i].find('#cite') != -1 or object1[i].find("geohack
```
2. בשדה של Area, חלק מהמדינות הכילו את הערך של השדות האלה באותה השורה של המילה Area וחלק מהן הערך היה שורה אחרי, ולכן היינו צריכים לחלק למקרים, תחילה מצאנו את המספר של השורה

שבו נמצאת המילה Area בעזרת השאילתה הזאת: `count(//table[contains(@class ,`
`"infobox")]/tbody/tr[contains(th/a//text(),"{}")]/preceding-sib`
`(*::ling`

ואז השתמשנו בשאילתה שמניחה שהם באותה שורה, ואז אם ערך החזרה היה ריק או מכיל רק זבל, אז הסקנו שזה מקרה קצה, ורק במקרה הזה הרצנו את השאילתה השניה שמניחה שהערך נמצא בשורה שאחרי המילה Area.

3. כאשר רוצים לחלץ עבור נשיאים/ראשי ממשלה את מיקום הלידה שלהם, נתקלים בבעיה שכן יש מעט ישויות עבורן מידע זה נשמר באיזור אחר בinfo box (זה נשמר בתוך תגית a)
על מנת לטפל בבעיה זו, הרצנו את השאילתה שתופסת את המקרה הנפוץ, ובמידה והיא מחזירה תשובה שגויה(בדקנו זאת באמצעות if כי ראינו שתמיד חוזרת תשובה זהה במקרה שהשאילתה הראשונה שמטפלת במקרה הנפוץ כושלת) אנחנו מריצים שאילתה נוספת שתופסת את המקרים המעטים שהשאילתה הראשונה אינה מצליחה לתפוס

השאילתה שהרצנו עבור המקרה הכללי:
`table[@class ="infobox//`
`[()vcard"]/tbody/tr[th//text()="Born"]/td/text()[last`

השאילתה עבור מקרה הקצה:
`table[@class ="infobox//`
`()vcard"]/tbody/tr[th//text()="Born"]/td/a[last()]/text`

אפשר לראות שבשאילתה שמתאימה למקרה הקצה הוספנו a בשאילתה מכיוון שלפעמים המידע היה שמור בתוך תגית a מה שלא קורה ברוב עמודי הויקיפדיה