# scrape_main

April 26, 2023

## 1 Measuring company similarities using text data

By: Filip Mellgren 2023

This is a "skills project" for the 2023 course in labor economics taught by Mitchell Downey and Horng Chern Wong at Stockholnm University. The idea is to practice a certain skill that can be useful in future research and is meant to be a small project where we get our hands dirty.

Hoberg, Phillips (2016), HP, pionered an approach to measure product similarities using text data and made a data set of such product similarities publicly available, see `http://hobergphillips.tuck.dartmouth.edu/`. For their data set, they make use of product descriptions available for public American corporations, the so called 10-K form fillings. To measure similarities, HP calculate word frequency and apply the cosine similarity algorithm. Higher values indicate closer (multi-dimensional) angle between two word representations, which means many words are shared between the two text inputs. If many words are shared, the idea is that two products share common features and are thus close substitutes.

Inspired by HP, I want to do something similar for Swedish firms. HP uses product desciptions, but for this project, I will use wikipedia pages containing company descriptions to estimate the matrix $\mathbf{A}'\mathbf{A}$ of cosine similarities. The project therefore starts with scraping data from Wikipedia, where my starting point is a Wikipedia page containing a table of Swedish firms with some descriptions. I scrape the full table, and then follow each individual link to download the Wikipedia page of each entry in that table. I save each Wikipedia page's text to a file for each entry in the table. I then use these saved files as input to the cosines similarity algorithm to calculate a number and populate a matrix. Finally, I do a simple visualization of the scraped data before concluding the project.

Further steps can follow the spirit of Bruno Pellegrino's Job Market Paper, where he uses the similarity matrix to estimate inverse demand and cross price elasticities of demand. In order to do that, he links the firms to price and quantity data, which I don't have, but this is a possible theoretical extension that may be feasible for industry specific studies where price data is available.

[ ]:
```
from selenium import webdriver
from selenium.webdriver.common.by import By
import time
import math
import re
from collections import Counter
import numpy as np
import pandas as pd
import os
```

```
import urllib.request
from bs4 import BeautifulSoup
import networkx as nx
import matplotlib.pyplot as plt
```

### 1.1 Starting point, scrape basic information

I will start at this Wikipedia page: `https://en.wikipedia.org/wiki/List_of_companies_of_Sweden`, which contains a list of many Swedish firms. It is by no means exhaustive but contains a decent number of companies and I think it is enough for this project.

I found this link: `https://alanhylands.com/how-to-web-scrape-wikipedia-python-urllib-beautiful-soup` which helps me scrape the Wikipedia table. So let's start from there to get the table into Python.

[ ]:
```
url = "https://en.wikipedia.org/wiki/List_of_companies_of_Sweden"
page = urllib.request.urlopen(url)
soup = BeautifulSoup(page, "lxml")
all_tables=soup.find_all('table', class_='wikitable sortable')

rows = all_tables[1].findAll('tr') # 'tr' = table row
header = rows[0]

colnames = ['link']
for col in header.findAll('th'):
    colnames.append(col.text.strip())

data_list = []
for row in rows:
    data_row = []
    try:
        link = row.find('a').get('href')
    except AttributeError:
        link = ''
    data_row.append(link)
    for col in row.findAll('td'):
        data_row.append(col.text.strip())
    data_list.append(data_row)

df = pd.DataFrame(data_list, columns = colnames)
df
```

[ ]:
```
                        link             Name           Industry  \
0                                          None               None
1       /wiki/3H_Biomedical    3H Biomedical        Health care
2     /wiki/AarhusKarlshamn  AarhusKarlshamn     Consumer goods
3        /wiki/Abba_Seafood     Abba Seafood     Consumer goods
4          /wiki/ABU_Garcia       ABU Garcia     Consumer goods
..                       …              …                …
```

```
298  /wiki/Wayne%27s_Coffee   Wayne's Coffee  Consumer services
299              /wiki/WESC             WESC     Consumer goods
300           /wiki/WG_Film          WG Film  Consumer services
301           /wiki/WM-data          WM-data         Technology
302    /wiki/X5_Music_Group   X5 Music Group  Consumer services

                          Sector Headquarters Founded  \
0                           None         None    None
1                  Biotechnology      Uppsala    2004
2                  Food products        Malmö    2005
3                  Food products   Gothenburg    1883
4          Recreational products  Svängsta[5]    1921
..                           …          …     …
298           Restaurants & bars    Stockholm    1994
299       Clothing & accessories    Stockholm    1999
300  Broadcasting & entertainment        Malmö    1994
301                     Software    Stockholm    1969
302  Broadcasting & entertainment    Stockholm    2003

                                      Notes
0                                      None
1                        Cell-based biotech
2                       Vegetable oils, fats
3                                   Seafood
4    Fishing reels, part of Newell Brands (US)
..                                       …
298                    Coffeehouse chain
299                             Clothing
300                       Film production
301               IT consulting, defunct 2008
302                       Music recordings

[303 rows x 7 columns]
```

Neat, the table contains a bunch of information that could potentially be useful. However, I care about the `link` column which contains links to various Wikipedia entries, which leads us to the next section.

### 1.2 Scrape text data

In this section, I scrape all wikipedia pages in the `link`column of the table above.

[ ]:
```python
def scrape_page(link):
    '''' Scrape the page of a given link and saves it as a .txt file.
    Input: link (str)
    Output: None

    '''
```

```
    url = "https://en.wikipedia.org/" + link

    if os.path.isfile(link[1:] + ".txt"):
        return
    time.sleep(0.3)
    driver.get(url)
    element = driver.find_element(By.ID , 'bodyContent')
    try :
        with open(link[1:] + ".txt", "w") as text_file:
            text_file.write(element.text)
    except FileNotFoundError:
        pass
    return
```

The function above creates a `url`, opens it up in a browser (these actions are visible on a PC as they happen in real time!), and saves the `bodyContent`, which is the Wikipedia text, into a file based on the `link` provided.

[ ]:
```
driver = webdriver.Chrome()
for i in range(1, df.shape[0]):
    link = df.iloc[i, 0]
    scrape_page(link)
```

We have now scraped all the links in our table and stored the Wikipedia text as .txt-files!

All company textfiles can be found under `wiki/` (not published to GitHub).

This is an example of the data we have:

[ ]:
```
with open("wiki/Atlet.txt", 'r') as f:
    text = f.read()
    print(text)
```

```
From Wikipedia, the free encyclopedia
(Redirected from Atlet)
Atlet AB
Founded 1958
Founder Knut Jacobsson
Headquarters Mölnlycke, Sweden
Area served 47 countries
Operating income SEK 1.8 billion
Number of employees around 1000
Website atlet.com
Atlet is a company that manufactures and markets indoor and outdoor trucks. The
company also provides services related to trucks and material handling, such as
logistics analysis, training and service. The head office, manufacturing and
training premises are located in Mölnlycke, just outside Göteborg, Sweden.
Atlet is a part of Nissan Forklift Co. Ltd., with subsidiaries in Belgium,
```

Denmark, France, Luxembourg, Norway, Sweden, the Netherlands, the UK and Germany. There are retailers in a further 36 countries.
History[edit]
Knut Jacobsson started Elitmaskiner in Göteborg in 1958.[1] At the time the company only made trucks for indoor use. Elitmaskiner changed its name to Atlet in 1966.[1]
The company started by making hand pallet trucks.[1] Around 1960, powered stackers and telereach trucks dominated the market. Knut Jacobsson then invented the pedestrian stacker that had a lifting capacity comparable with the telereach trucks, but could be used in narrower aisles, thanks to its patented side stabilizers.[1]
Atlet started providing training for truck operators in the 70s. Technical developments continued with computerized simulations of warehouse management solutions, automatic trucks and mobile terminal systems, and between 1988 and 1994 Atlet took part in a development project in collaboration with doctors and occupational therapists. This resulted in Tergo, a telereach truck with ergonomic solutions such as the mini steering wheel and floating armrest.
Jacobsson left the CEO position in 1995, letting his daughter, Marianne Nilson, take over.[2] It was then Sweden's biggest family-owned engineering company and one of the leading European truck manufacturers.[2]
Nissan Forklift, a subsidiary of the Nissan Motor Company, bought Atlet AB in 2007.[3] Nissan Motor Company spun-off its Industrial Machinery Division and establish a new company, "Nissan Forklift Co., Ltd.", effective from October 1, 2010.
[4]
See also[edit]
A Ergo
Sources[edit]
^
a b c d Bergqvist, Peter. "Bra affärer för Atlet", Verkstäderna. 28 November 2012, p. 22.
^
a b Börjesson, Karin. "Atlets nya VD har tagit över", Göteborgs-Posten. 28 April 1995.
^ "Nissan tar över Atlet", Göteborgs-Posten. 5 September 2007.
^ Atlet, Henrik Moberger, Tärnan Reportage AB 2008. ISBN 978-91-633-1924-2
External links[edit]
Atlet AB
Categories: Manufacturing companies of SwedenNissanCompanies based in Västra Götaland County

We can see that we get the summary description followed by the text description of the company.

Parts of the text are likely to be shared among all entries, such as `From Wikipedia, the free encyclopedia`.

### 1.3 Calculate similarity score

Here, we use cosine similarity algorithm to define how similar two bodies of text are and apply it to our scraped Wikipedia data.

[ ]:
```
WORD = re.compile(r"\w+")

def get_cosine(vec1, vec2):
    ''' Calculate cosine similarity between two vectors.
    Most code comes from:
    https://stackoverflow.com/questions/15173225/
 ↪calculate-cosine-similarity-given-2-sentence-strings
    Input: vec1, vec2 (dict)
    Output: cosine similarity (float)
    '''
    intersection = set(vec1.keys()) & set(vec2.keys())
    numerator = sum([vec1[x] * vec2[x] for x in intersection])

    sum1 = sum([vec1[x] ** 2 for x in list(vec1.keys())])
    sum2 = sum([vec2[x] ** 2 for x in list(vec2.keys())])
    denominator = math.sqrt(sum1) * math.sqrt(sum2)

    if not denominator:
        return 0.0
    else:
        return float(numerator) / denominator

def text_to_vector(text):
    ''' Convert text to vector.
    Input: text (str)
    Output: vector (dict)
    '''
    words = WORD.findall(text)
    return Counter(words)

def file_to_vector(file):
    ''' Convert file to vector.
    Input: file (str) path to a file
    Output: vector (dict)
    '''
    try:
        with open(file, 'r') as f:
            text = f.read()
    except FileNotFoundError:
        print("path not found")
        return
    return text_to_vector(text)
```

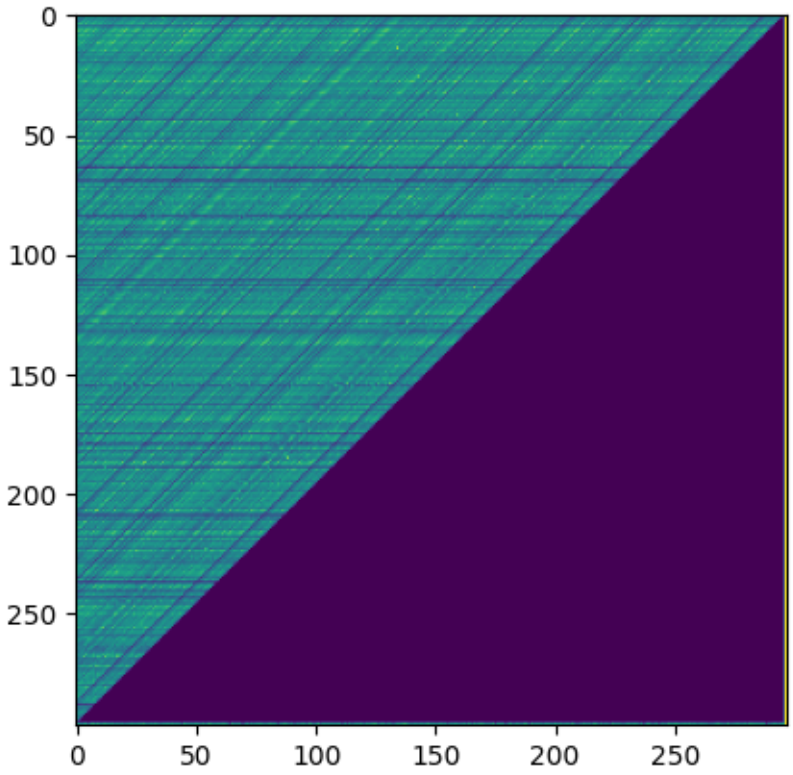
With the functions in place, we can use them to loop over all pairs of files, and thereby calculate the mesure we care about, the cosine distance metric.

The nested for loop algorithm is a little slow, it should take ca 30 seconds to run.

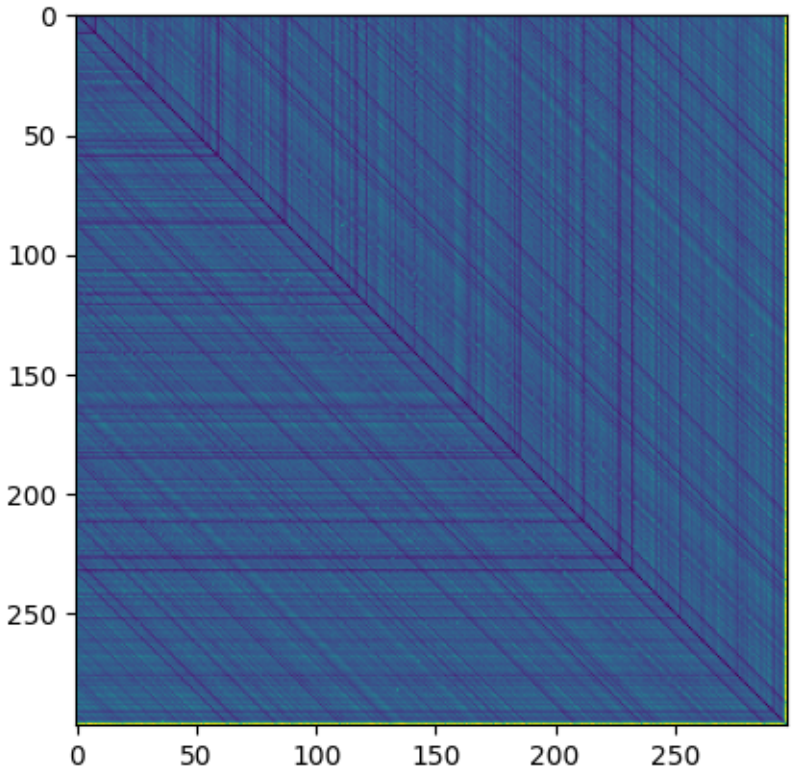
[ ]:
```
# Loop over all files in wiki folder
n_files = len(os.listdir('wiki'))
distances = np.zeros((n_files-1, n_files-1))
for count1, file in enumerate(os.listdir('wiki')):
    remaining = os.listdir('wiki')[count1:]
    file = 'wiki/' + file
    vector1 = file_to_vector(file)
    for count2, file2 in enumerate(remaining):
        file2 = 'wiki/' + file2
        vector2 = file_to_vector(file2)
        cosine = get_cosine(vector1, vector2)
        distances[count1 - 1, count2 - 1] = cosine
```

We now have the following matrix. Notice how only half is populated. Because of symmetry, we know what the other half will look like, so let's turn this matrix around and impose symmetry on it.

[ ]:
```
plt.imshow(distances)
plt.show()
```



[ ]:
```
distances = np.flipud(distances)
distances = distances + distances.T - np.diag(distances.diagonal())
plt.imshow(distances)
plt.show()
```



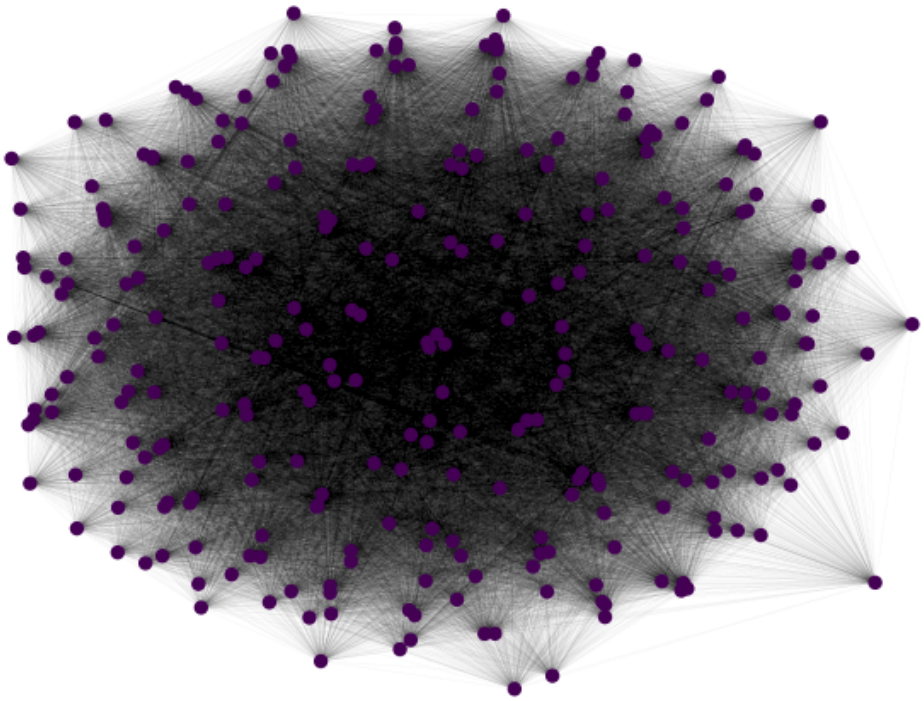

[ ]:
```
n = np.shape(distances)[0]
# Create graph from distance matrix
G = nx.Graph()
for i in range(n):
    for j in range(i+1, n):
        G.add_edge(i, j, weight=distances[i, j])

# Compute positions of nodes using Fruchterman-Reingold algorithm
pos = nx.spring_layout(G, seed=42)

# Draw graph with nodes colored by their degree centrality
degree_centrality = nx.degree_centrality(G)
node_color = [degree_centrality[i] for i in G.nodes()]
node_size = [25*degree_centrality[i] for i in G.nodes()]
```

```
nx.draw(G, pos, node_color=node_color, node_size=node_size, with_labels=False,␣
 ↪width = 0.01)

# Show the plot
plt.show()
```



And there we have it! It seems there are no specific clusters in this data and that everything is rather uniform. Perhaps because we did not spend time cleaning the Wikipedia files of elements that are common across all articles.

For future purposes, I think it makes sense to evaluate the text description and use product data as in the seminal HP work. This exercise also highlights the importance of cleaning the text data.