Academic Paper Recommender

Nasim Shahbazian
Capstone Project 2
Springboard Data Science
November 2018

The Problem

- The aim of this project is to create a recommender system to help researchers and scientists find related articles to a specific paper.

- There are thousands of papers published everyday and it is not feasible for a researcher to browse all these papers in order to find a related paper to a desired subject.

- This recommender system helps scientists with their search and saves them a lot of time looking into article resources.

Dataset

- The data is collected using API for PubMed, a repository for biomedical data.
- The data is extracted in raw XML in a full text format and information such as paper’s ID, title, authors last name, year, journal, abstract, tags, and citations is collected from each paper.
- Programmatically queried via the NCBI Entrez E-utilities interface.

Data Scraping

- Using Entrez Programming Utilities (E-utilities) at the National Center for Biotechnology Information (NCBI).

- from Bio import Entrez
- from Bio.Entrez import efetch

```
def create_idlist(n):
    """Return a list of PMID numbers with size n. The earliest paper PMID is 28785052 published on August 2017"""
    idlist = []
    for i in range(27000009-n, 27000009):
        idlist.append(str(i))
    return idlist
```

```
def fetch_details(id_list):
    """A function that gets a list of PMIDs and returns a xml of papers' information"""
    ids = ','.join(id_list)
    handle = Entrez.efetch(db='pubmed',
                           retmode='xml',
                           id=ids)
    results = Entrez.read(handle)
    return results
```

Data Scraping

```python
def get_title(paper):
    """Given a xml paper info, this function returns the paper's title"""
    return paper['MedlineCitation']['Article']['ArticleTitle']
```

```python
def get_abstract(paper):
    """Given a xml paper info, this function returns the paper's abstract"""
    try:
        return paper['MedlineCitation']['Article']['Abstract']['AbstractText'][0]
    except:
        return np.nan
```

```python
def get_year(paper):
    """Given a xml paper info, this function returns the paper's year of publication"""
    try:
        return paper['MedlineCitation']['Article']['Journal']['JournalIssue']['PubDate']['Year']
    except:
        return np.nan
```

```python
def get_journal(paper):
    """Given a xml paper info, this function returns the paper's journal name"""
    try:
        return paper['MedlineCitation']['Article']['Journal']['Title']
    except:
        return np.nan
```

Data Scraping

- The the Dataframe is created as follows:

```
def get_paper_info(paper, id):
    """Given paper's xml and PMID, it returns a tuple containing infomration about the paper"""

    title = get_title(paper)
    authors = get_authors(paper)
    tags = get_tags(paper)
    citations = get_citations(paper)
    year = get_year(paper)
    abstract = get_abstract(paper)
    journal = get_journal(paper)
    return (id, title, authors, year, journal, abstract, tags, citations)
```

```
if __name__ == '__main__':
    id_list = create_idlist(10000)
    print(len(id_list))
    try:
        papers = fetch_details(id_list)
    except:
        pass

    paper_list = []

    for i, paper in enumerate(papers['PubmedArticle']):
        #if i==20:
        #    print (paper)
        paper_list.append(get_paper_info(paper, id_list[i]))

    df = pd.DataFrame(paper_list, columns=['id', 'title', 'authors', 'year', 'journal', 'abstract', 'tags', 'citations'
```

Resulting Dataframe

	id	title	authors	year	journal	abstract	tags	citations
0	26990009	Identifying Older Adults with Serious Illness:...	[Kelley, Covinsky, Gorges, McKendrick, Bollens...	2017	Health services research	To create and test three prospective, increasi...	[Activities of Daily Living, Aged, Aged, 80 an...	[15493448, 17187548, 23838378, 9441588, 198285...
1	26990010	Social rank versus affiliation: Which is more ...	[Wang, Sun, Sheeran, Sun, Zhang, Zhang, Xia, Li]	2016	American journal of primatology	Research on leadership is a critical step for ...	[Animals, Grooming, Leadership, Macaca, Moveme...	NaN
2	26990011	Three-dimensional manometry of the upper esoph...	[Meyer, Jones, Walczak, McCulloch]	2016	The Laryngoscope	High-resolution manometry (HRM) is useful in i...	[Adult, Deglutition, Deglutition Disorders, Es...	[10718434, 23728150, 16410365, 17305278, 44782...

Data Cleaning

- Since the search is going to be based on title and/or abstract and also tags, the papers with missing title and tags were removed from the data.

- We did not remove papers which have title but no abstract. In this case, only title is used to find similar papers.

```
df.dropna(axis=0, subset=['title','tags'], inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7690 entries, 0 to 9631
Data columns (total 8 columns):
id           7690 non-null object
title        7690 non-null object
authors      7554 non-null object
year         7315 non-null object
journal      7690 non-null object
abstract     6806 non-null object
tags         7690 non-null object
citations    2656 non-null object
dtypes: object(8)
memory usage: 540.7+ KB
```

Data Cleaning

- The .csv file dataset is uploaded in Databricks, which is a web-based platform for working with Spark.
- The full data is uploaded in the Databricks as an RDD.
- The punctuation like commas and quotes are removed from the text (string).
- Keeping contractions together.
- The method also makes the words lower cased.

Data Cleaning

```
def remove_punctuation(text):
    """ This method removes the punctuation like commas and quotes from the text (string).
    We also want to keep contractions together. The method also make the words lower cased.
    It returns a list or words in the text
        Args:
            text (string): the text we want to clean
        Return:
            A list with cleaned words
    """
    # split into words by white space
    words = text.split()
    words_lower = [w.lower() for w in words]

    # Remove punctuation from each word
    table = str.maketrans('', '', string.punctuation)
    stripped = [w.translate(table) for w in words_lower]
    return ' '.join(stripped)
```

Data Cleaning, Tokenizing & Stopwords

- Tokenizer and StopWordsRemover from pyspark.ml.feature are used to clean the data.

- Tokenization is the process of taking text and breaking it into individual terms (usually words).

- Stop words are words which should be excluded from the input, typically because the words appear frequently and don’t carry as much meaning. StopWordsRemover takes as input a sequence of strings and drops all the stop words from the input sequences.

Modeling Approaches

- Text similarity-based recommender: Title and abstract of each paper is used. TF-IDF based similarity is calculated to recommend the n number of related papers.

- Semantic similarity based recommender: The tags are used for determine similarity between papers. Papers which are sharing the most number of tags with the reference paper would be recommended.

- The recommendation algorithms are implemented in Spark using Python, and are run using the web-based platform, Databricks, on their provided automated cluster.

TF-IDF

$$IDF(t, D) = \log \frac{|D| + 1}{DF(t, D) + 1},$$

$$TFIDF(t, d, D) = TF(t, d) \cdot IDF(t, D).$$

- TF: HashingTF from spark.ml is used to generate the term frequency vectors.

- IDF: An estimator which is fit on a dataset and produces an IDFModel. The IDFModel takes feature vectors and scales each feature. It also down-weights frequently appeared features

Normalizer and Cosine Similarity

- The output of the TF-IDF is then passed through a Normalizer. Normalizer transforms a dataset of vector rows, normalizing each vector to have unit norm.

- Cosine Similarity of the vectors is then calculated using the cartesian product and the function dot on numpy arrays to produce a similarity array between each pair of papers in the whole dataset. .

Finding Similarity Arrays

```
# get the similarities for each pair of papers
def get_similarities(paper_rdd, paper_tfidf):
    """ Function that returns the array of similarities between each two papers
        Args:
            paper_rdd (rdd): idd of all papers abstract and titles
            paper_tfidf (pyspark.sql.dataframe.DataFrame): tf-idf vectors for a given paper
        Return:
            similarity_array:  array of cosine similarities for each pair of papers
    """

    print("... Computing L2 norm ...")
    labels = paper_rdd.map(lambda x: x[0])
    features = paper_tfidf

    normalizer = Normalizer(inputCol="features", outputCol="normFeatures")
    data = labels.zip(normalizer.transform(features).rdd.map(lambda r: r.normFeatures))

    #Using a Cartesian product and the function dot on numpy arrays:
    similarity_array = data.cartesian(data)\
    .map(lambda l: ((l[0][0], l[1][0]), l[0][1].dot(l[1][1])))\
    .sortByKey()

    return similarity_array
```

Finding n Neighbors

```
# get the n top similar papers for given paper info
def get_neighbors(paper_PMID, similarity_array, n):
    """ Function that returns the 50 most similar papers for given paper
        Args:
            paper_PMID (int): PMID of the paper we want to find similar papers to
            similarity_array (array): cosine similarity array for all papers
            n (int): number of similar papers we are looking for, for a specified paper
        Return:
            list: the list of papers relevant to the given paper based on cosine similarity
    """
    candidates = similarity_array.filter(lambda x: x[0][0]==paper_PMID).sortBy(lambda a: -a[1])
    neighbors = candidates.map(lambda x: x[0][1])

    return neighbors.take(n)
```

Results, Title/Abstract-based Similarity

- Main paper title: 'Glucose Metabolism After Gastric Banding and Gastric Bypass in Individuals With Type 2 Diabetes: Weight Loss Effect.'

- First two recommended papers:
- 1- 'Laparoscopic sentinel node navigation surgery for early gastric cancer: a prospective multicenter trial.'
- 2- 'Can lymphovascular invasion be predicted by preoperative multiphasic dynamic CT in patients with advanced gastric cancer?'

Results, Tags-based Similarity

- Main paper tags: ['Adult', 'Bariatric Surgery', 'Diabetes Mellitus, Type 2', 'Female', 'Gastric Bypass', 'Glucagon-Like Peptide 1', 'Glucose', 'Humans', 'Incretins', 'Insulin Resistance', 'Longitudinal Studies', 'Male', 'Middle Aged', 'Obesity', 'Postoperative Period', 'Prospective Studies', 'Sweetening Agents', 'Weight Loss']

- First two recommended papers:

- 1- ['Coronary Artery Disease', 'Diabetes Complications', 'Diabetes Mellitus, Type 1', 'Diabetes Mellitus, Type 2', 'Diabetic Cardiomyopathies', 'Glycated Hemoglobin A', 'Heart Failure', 'Humans']

- 2- ['Animals', 'Diabetes Mellitus, Experimental', 'Fibroblast Growth Factors', 'Glucagon', 'Glucagon-Like Peptide 1', 'Glucuronidase', 'Hyperglycemia', 'Insulin', 'Islets of Langerhans', 'Male', 'Mice', 'Mice, Inbred C57BL', 'Mice, Transgenic', 'Streptozocin']

Acknowledgements

- My mentor, Dipanjan Sarkar
- Springboard
- Course teaching assistant, Andrea Constantinof