Recommendation System on DBLP Dataset (Experiment 1)

How to use it

1. Code can be found on GitHub: https://github.com/weakties/infrastructure/tree/master/component_DBLP

```
xxxx:component_DBLP zzzz$ ls
HICSS                                   getAuthor.py   hicssAuthorContents.py
RecommendationSys_DBLP_Implementation.pages getConf.py list
example.py                              getTitle.py    recommendationsys.py
extractconferencedata.sh                getYear.py
```

2. Download DBLP data from http://dblp.uni-trier.de/xml/ (dblp.xml.gz and dblp.dtd) to directory "./DBLPdataset". If there is no such directory, create one

```
xxxx:component_DBLP zzzz$ mkdir DBLPdataset
xxxx:component_DBLP zzzz$ cd DBLPdataset
xxxx:component_DBLP zzzz$ mv ~/Download/dblp.xml.gz ~/Download/dblp.dtd ./
```

3. Run the recommendation system against certain user on the given conferences

   Single user:
   python example.py username outputDir conference1 conference2 … conferenceN

   Example:

```
xxxx:component_DBLP zzzz$ python example.py "Jari Kangas" Jari  "CHI Extended
Abstracts"  ETRA CHI
```

   It will run the recommendation system against user "Jari Kangas" on the conferences "CHI Extended Abstracts" ETRA and CHI. The results will be put in the directory "Jari"

   Batch:
   python example.py list

   Example:

```
xxxx:component_DBLP zzzz$ python example.py list
xxxx:component_DBLP zzzz$ cat list
Aino Ahtinen,Aino,NordiCHI,INTERACT,MUM,Mobile HCI,AmI,OZCHI
Heli Väätäjä,Heli,MindTrek,OZCHI,CHI Extended Abstracts,NordiCHI,Mobile HCI
Harri Siirtola,Harri,IV,CHI Extended Abstracts,INTERACT
```

   Put all the users and their conferences in a list file, one for each line. The format is as following:

   user name,outputDir,Conference 1,Conference 2,Conference 3,…,Conference N

   NOTE: There is no space between each component but only a comma. It is allowed to contain space within the component, for example, the user name and conference name

Implementation Details

1. Data collection and clean

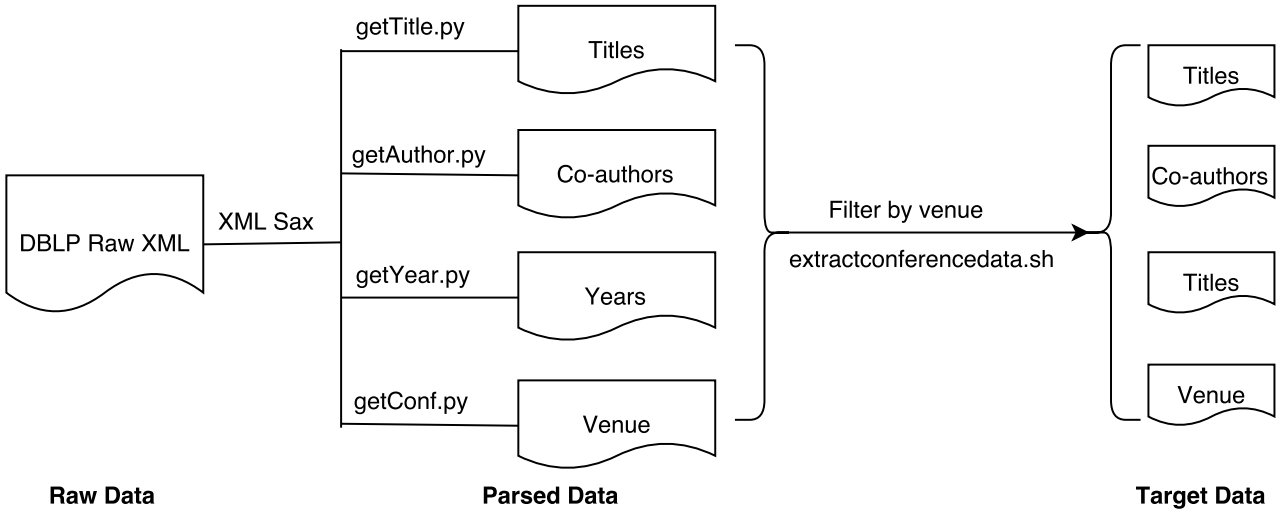

Figure 1.1

The raw DBLP data is a huge XML file (more then 2 GB). Firstly, the DBLP raw XML file is parsed by a group python scripts (getXXX.py) to extract the title, co-authors, publish year and venue from all the articles. After breaking down the raw DBLP data, a subset data is extracted by user defined venues which is the target dataset for the recommendation system.

2. Initialisation

The target data are saved in four text files and each file should have the same number of records N:
File titles.txt contain N publication titles.
File co-authors.txt contains all the co-authors in each publication.
File years.txt contains the year of each publication.
File booktitles.txt contains the venue of the publication.

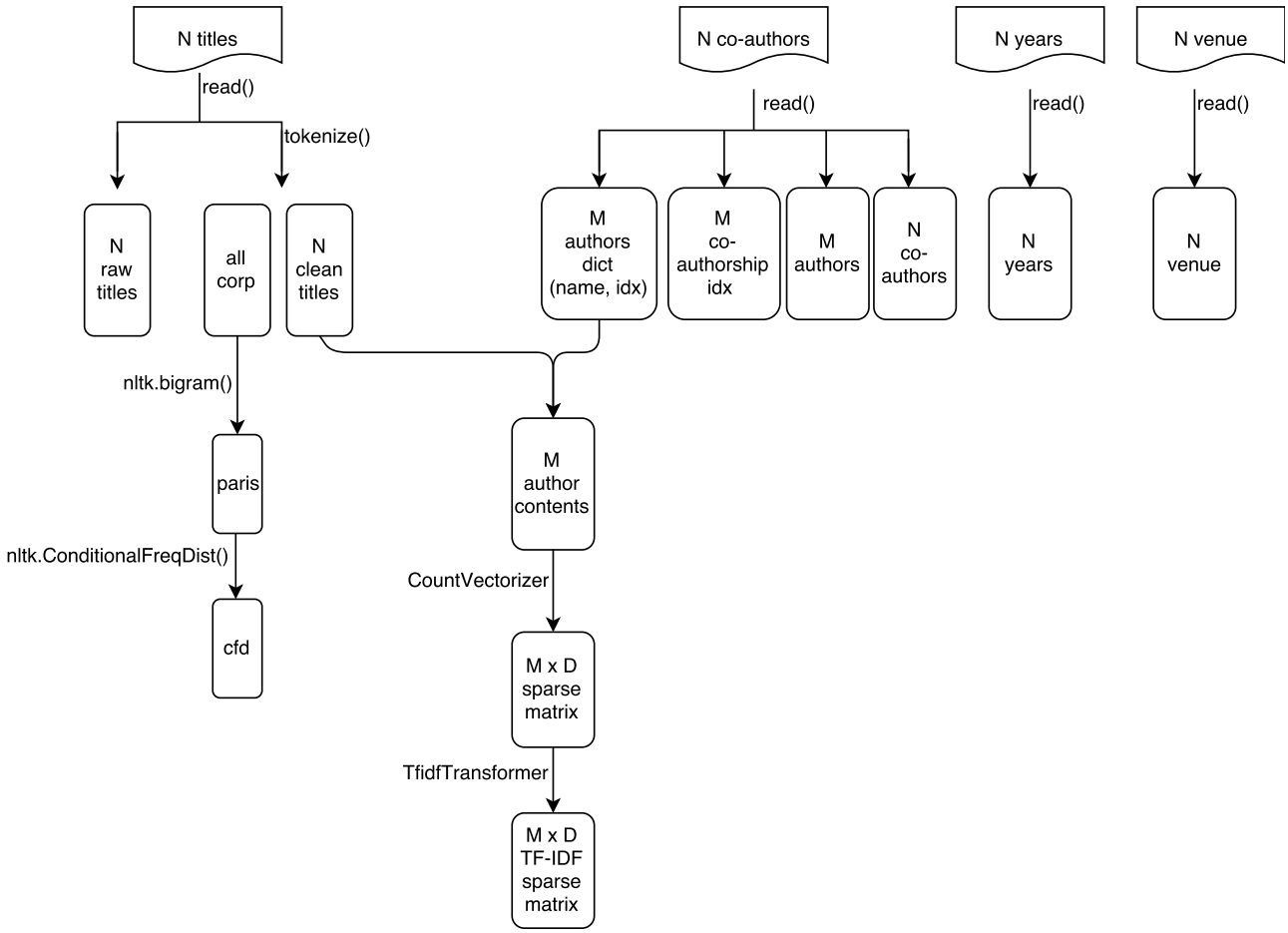

Figure 2.1

The “raw titles” are the original titles read from the titles.txt. It will be listed in the recent publication of a recommended author.

The ‘all corp’ is from the tokenisation of all the publication titles. It is passed to nltk.bigram() to create bigram pairs. All the pairs are used to create the conditional frequent distribution (cfd) for each word. The cfd is used to generate meaningful bigram research topics of an author based on all his/her publication titles.

The ‘clean titles’ contains the content after removing stop words and digitals.

The ‘co-authors’ is all the records in the file co-authors.txt

The ‘authors dict’ is a dictionary to store all the authors which is more than the number of co-authors records (M >= N). The reason of using dictionary rather than list is to speed up search the index by name. The key of the dictionary is the author’s name and the value is his/her idx in other lists which store other information of the author, such as co-authorship, all publications,

TF-IDF vector, etc.

At the same time of constructing the authors dictionary, the ‘author contents’ is keeping updated with new content of the author from ‘clean titles’. The ‘author contents’ is the key part of the system which will be vectorized and fit into the TF-IDF transformer to create the TF-IDF feature vector of each author. All the TF-IDF feature vectors as a whole is a M x D sparse matrix. D is the dimension of the TF-IDF feature vector.

The ‘authors’ is a list of all the authors.

The ‘co-authorship idx’ stores the idx of each co-authors in the ‘authors’ list, for example, one author has the co-authorship idx of [0, 1, 2, 1, 4, 6, 7, 4], then the name of his/her co-authors can be found in the ‘authors’ list by indexes [0, 1, 2, 4, 6, 7] and also the number of co-authorship can be calculated: 2 times co-authorship with author of index 1 and 4, one time co-authorship with the others.

The ‘years’ stores the corresponding publication year of each publication.

The ‘venue’ stores the conference where each publication is published. In DBLP XML file, the tag for all the conferences is ‘booktitles’.

3. Run recommendation

A user name and target conferences are required to run the recommendation system. Firstly, the index of the user is retrieved from ‘authors dict’, and then the TF-IDF feature vector of the user (*1xD* TF-IDF vector) is extracted from the *M x D* TF-IDF sparse matrix. Next, pairwise_distances() is called to compute the distances between the user and all the other users based on their TF-IDF feature vectors. The valid metrics of the distance in function pairwise_distances() are [‘cityblock’, ‘cosine’, ‘euclidean’, ‘l1’, ‘l2’, ‘manhattan’], and ‘euclidean’ is applied by default. The distance array is further sorted and the original position of the index is also stored by argsort(). The size of the distance array is *M* as there are totally *M* authors, include user himself/herself.

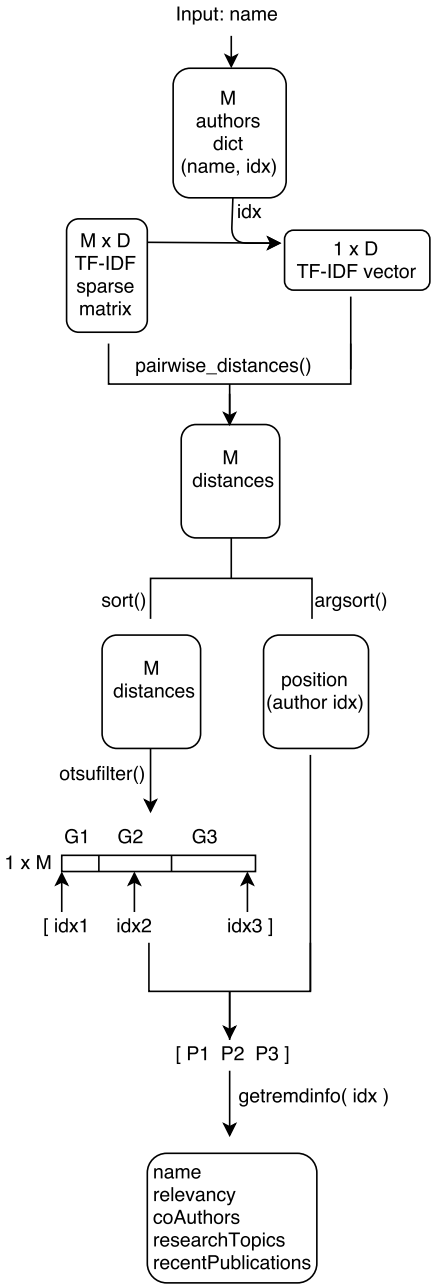

Figure 3.1

In the experiment 1, the recommendations are chosen from three groups (G1, G2, G3): most relevant, medium and not relevant at all. The separation is based on the similarity distance. In order to find the threshold distance for each group, the otsu filter (https://en.wikipedia.org/wiki/

Otsu%27s_method) is employed. The otsu filter will return the position where minimizes the intra-class variance (Figure 3.2). As we want to have 3 groups, so we applied otsu filter twice:

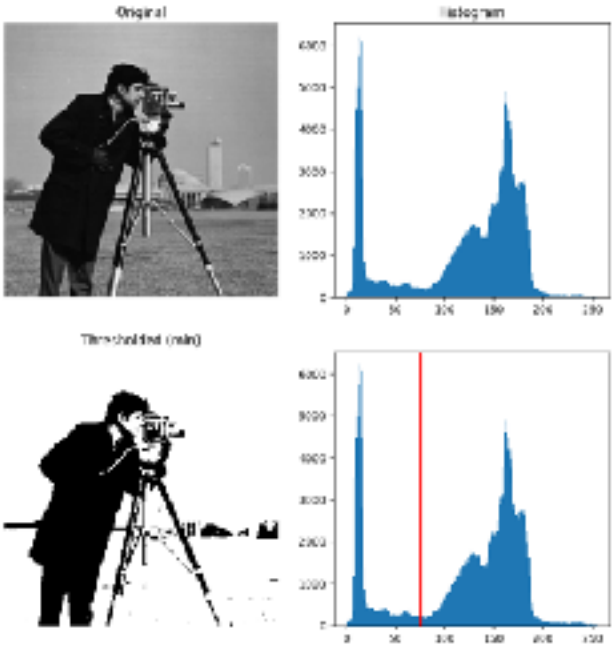

Figure 3.2

firstly, we apply the otsu filter on the whole similarity distance array to separate the distances into two parts. After that, we apply otsu filter again only on the second half. Eventually, we have two threshold and three groups of distances.

In this experiment, in order to better represent each group recommendations, the top 3 (idx 1) most similar candidates (based on the similarity distance) are chosen to represent the most relevant ones (G1). Another 3 from the middle (idx 2) of the second group are chosen to represent the medium relevant ones (G2). The last 3 (idx 3) are chosen to represent the ones not relevant at all (G3).

The [idx1, idx2, idx3] are the indexes in the sorted distance array. The original positions [P1, P2, P3] of the [idx1, idx2, idx3] in the result of argsort() are used to index those authors.

The detail information of each recommendation is generated by function getremdinfo(). Each recommendation has the following information: name, relevancy, co-authors, research topics (by using the nltk.biagram and conditional frequent distribution), recent publications.