ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Advanced Topics in Data Engineering
Assignment 2

Name : Ilias Dimos
AM : f2822102
Professor : Georgios Alexiou

TASKS

A. Use the **Token Blocking** (**not to be confused with Standard Blocking**) method to create blocks in the form of K-V (Key-value) pairs. The key for every entry will be each distinct Blocking Key (*BK) derived* from the entities' attribute values and the values for each *BK* will be the entities' ids. Please note that the id column in the data can be used only as reference for the blocking index and it will NOT be used in the blocking process (block index creation). Please also note that you are advised to transform every string to **lower case** during the tokens' creation (before you insert it in the index) to avoid mismatches. **At the end of the creation use a function to pretty-print the index.**

Solution

Dataset Cleaning

In order to use the **Token Blocking** method, we had to clean our data first and make some transformations also we made the index of the dataset begin from one instead of the zero which is the default.

To begin with , we made all the strings of the dataset lower case to avoid any mismatches during the token's creation.

Secondly , the English stop words have been removed. That why because they are words with high frequency use so each one of them would have been contained in a lot of tokens.

It has been observed that a lot of strings have been typed wrong and had some punctuations, so we removed them.

Finally , the **nan** values of the original dataset have been replaced with the word 'nan' in order to make them strings and remove them in the below mentioned procedure.

After the cleaning of the dataset the final from is :

	id	authors	venue	year	title
1	1	qd inc	san diego	nan	11578 sorrento valley road
2	2	argon jg hannoosh	phil mag	nan	initiation crazes polystyrene
3	3	gh hansen wetterberg h sjãstrãm norãn	histochemical journal	1992	immunogold labelling quantitative method demon...
4	4	tm hammett p harmon w rhodes	see	nan	burden infectious disease among inmates releas...
5	5	jr cogdell	new directions teaching learning	1995	role faculty advising science engineering
...	...	...	...	...	...
66875	66875	shukla p deshpande j naughton k ramasamy	vldb	1996	storage estimation multidimensional aggregates...
66876	66876	nan	nan	2003	call book reviews
66877	66877	r ramakrishnan ram	vldb	1996	modeling design versions
66878	66878	j shafer r agrawal mehta	vldb	1996	sprint scalable parallel classifier data mining
66879	66879	nan	nan	2003	report first international conference ontologi...

66879 rows × 5 columns

Figure 1 : Final Cleaned Dataset

The next step is to transform the dataset into a dictionary. We choose dictionary over a list because the values of the dictionary can be read much faster than in a list. After that we

removed the “nan” value mentioned before. The first 5 elements of the final dictionary can be seen in the figure 2 below :

```
{1: ['qd', 'inc', 'san', 'diego', '11578', 'sorrento', 'valley', 'road'],
 2: ['argon', 'jg', 'hannoosh', 'phil', 'mag', 'initiation', 'crazes',
     'polystyrene'],
 3: ['gh', 'hansen', 'wetterberg', 'h', 'sjãstrãm', 'norãn', 'histochemical',
     'journal', '1992', 'immunogold', 'labelling', 'quantitative', 'method',
     'demonstrated', 'studies', 'aminopeptidase', 'n'],
 4: ['tm', 'hammett', 'p', 'harmon', 'w', 'rhodes', 'see', 'burden',
     'infectious', 'disease', 'among', 'inmates', 'releasees', 'correctional',
     'facilities'],
 5: ['jr', 'cogdell', 'new', 'directions', 'teaching', 'learning', '1995',
     'role', 'faculty', 'advising', 'science', 'engineering']}
```

Figure 2 : First 5 Elements of the final dictionary

As we can see in figure 2, we accomplished to create a dictionary with every single word of each row. Now we are ready for the token blocking procedure.

Token Blocking

The idea behind token blocking is to find the id’s where every unique word belongs to by creating key – value pairs. In our case token corresponds to every unique word of the dataset and the keys are the IDs of every entity that we can find the specific token.

To begin with the token blocking procedure , two empty dictionaries have been initialized. The first one named “blocks” is used to store every token along with its ids. The second one named “unique_tokens” is used as storage of every unique word.

The idea we relied on was to iterate over the items of the dictionary in figure 2. In every iteration we check if the specific word is contained in the dictionary “unique_tokens”. If it is true then we add the index (of the dictionary not of the data frame) of the specific word , if it is false, we add the new word and the index as we did before.

After the completion of the procedure, we keep only the blocks with more than 2 entities. In the figure 3 the outcome of the token blocking can be observed.

```
{'qd': [1, 55360],
 'inc': [1, 297, 471, 852, 890, 923, 2857, 3057, 3486, 3486, 3599, 3766, 3766,
         4219, 4378, 4449, 4476, 4854, 4995, 5438, 5589, 6339, 6339, 6429, 6839,
         7038, 8574, 8574, 9238, 9368, 9368, 9377, 9927, 10500, 11596, 11892,
         13149, 14166, 14251, 14930, 15004, 15272, 15454, 15468, 15478, 15886,
         16358, 17005, 17983, 18160, 18337, 18337, 19281, 20115, 20153, 21912,
         22216, 22275, 22639, 23987, 24308, 25060, 25062, 25577, 25834, 25950,
         26475, 27244, 27539, 27714, 27806, 28545, 28981, 29327, 29690, 30987,
         31441, 31529, 31899, 32034, 32387, 32596, 33261, 34459, 34677, 34680,
         34840, 35677, 36229, 36256, 36411, 36661, 36797, 36949, 37178, 38311,
         38590, 38700, 39000, 39570, 39716, 39796, 40028, 40028, 40040, 40070,
         40372, 41393, 42111, 42685, 42867, 43028, 43073, 43180, 43598, 43918,
         43918, 44647, 44775, 44799, 44825, 44908, 45496, 45497, 45543, 45758,
         45843, 45994, 46763, 47124, 47909, 49164, 49406, 50987, 51722, 51861,
         51988, 52326, 52505, 53149, 53195, 53497, 54498, 55216, 55869, 55871,
         56525, 56545, 56805, 56890, 57202, 57323, 58275, 58857, 58936, 58981,
         59169, 60213, 60375, 60442, 61135, 61288, 61397, 61770, 62385, 62978,
         65238, 66194],
```

Figure 3 : Token Blocking procedure outcome for the first 2 tokens.

As we can see we found that the word **'qd'** has been found in the entities with ID **1** and **55360**.

B. Compute all the possible comparisons that shall be made to resolve the duplicates within the blocks that were created in Step A. **After the computation, please print the final calculated number of comparisons.**

Solution

In the step B we created a procedure that is running through the blocks and computes the comparisons using the formula **n(n-1)/2** where **n** is the length of each block , in every iteration we store the outcome in the "comparisons" variable summing the outcome with the previous ones.

At the end of the procedure, we have the final number of comparisons we need to make to resolve the duplicates. The number of comparisons needed to resolve the duplicates is : **546,651,578**.

C. Create a **Meta-Blocking graph** of the block collection (created in step A) and using the **CBS** Weighting Scheme (i.e., Number of common blocks that entities in a specific comparison have in common) **i) prune** (delete) the edges that have **weight < 2 ii) re-calculate** the final number of comparisons (like in **step B**) of the new block collection that will be created **after** the edge pruning

Solution

The main idea behind the Meta Block is the restructure of a redundancy-positive block collection into a new one that contains a substantially lower number of comparisons, while being equally effective.

The more blocks two entities share the more similar and the more likely they are to be matching.

Preparation

In our case the nodes of the graph will be the entities and for every pair of co-occurring entities we will add an undirected edge. Because of the large computational time it is needed to compute the pairs we created a function that the user can add the number of blocks he wants to slice and compute their pairs.

In order to run this task successfully we created a data frame with two columns. The first column is the “Token” and the second one is the “ID” which contains all the ids where we can found the specific token. In figure 4 we can see the format of the data frame :

	Token	ID
0	qd	[1, 55360]
1	inc	[1, 297, 471, 852, 890, 923, 2857, 3057, 3486,...
2	san	[1, 8, 11, 65, 215, 219, 352, 382, 496, 574, 6...
3	diego	[1, 8, 11, 65, 215, 219, 352, 382, 574, 579, 7...
4	valley	[1, 6357, 7798, 8932, 8993, 15672, 16285, 2171...
...	...	...
38346	clauen	[65844, 66183, 66659, 66746]
38347	muoz	[66026, 66788]
38348	schning	[66033, 66309]
38349	khn	[66109, 66308]
38350	cudrmauroux	[66215, 66858]

38351 rows × 2 columns

Figure 4 : Token – ID Data frame

In Meta Blocking we are not interest in Token but in the IDs, so in order to find the pairs we transformed the “ID” column of the figure 4 into a list (figure 5).

```
[[1, 55360],
 [1, 297, 471, 852, 890, 923, 2857, 3057, 3486, 3486, 3599, 3766, 3766, 4219,
  4378, 4449, 4476, 4854, 4995, 5438, 5589, 6339, 6339, 6429, 6839, 7038, 8574,
  8574, 9238, 9368, 9368, 9377, 9927, 10500, 11596, 11892, 13149, 14166, 14251,
  14930, 15004, 15272, 15454, 15468, 15478, 15886, 16358, 17005, 17983, 18160,
  18337, 18337, 19281, 20115, 20153, 21912, 22216, 22275, 22639, 23987, 24308,
  25060, 25062, 25577, 25834, 25950, 26475, 27244, 27539, 27714, 27806, 28545,
  28981, 29327, 29690, 30987, 31441, 31529, 31899, 32034, 32387, 32596, 33261,
  34459, 34677, 34680, 34840, 35677, 36229, 36256, 36411, 36661, 36797, 36949,
  37178, 38311, 38590, 38700, 39000, 39570, 39716, 39796, 40028, 40028, 40040,
  40070, 40372, 41393, 42111, 42685, 42867, 43028, 43073, 43180, 43598, 43918,
  43918, 44647, 44775, 44799, 44825, 44908, 45496, 45497, 45543, 45758, 45843,
  45994, 46763, 47124, 47909, 49164, 49406, 50987, 51722, 51861, 51988, 52326,
  52505, 53149, 53195, 53497, 54498, 55216, 55869, 55871, 56525, 56545, 56805,
  56890, 57202, 57323, 58275, 58857, 58936, 58981, 59169, 60213, 60375, 60442,
  61135, 61288, 61397, 61770, 62385, 62978, 65238, 66194],
```

Figure 5 : List of the IDs that will be used in Meta Blocking Procedure

Meta Blocking

Now we are ready to compute the edge pairs with their weights. As we said before the time for the implementation of the Meta Blocking Procedure is extreme time consuming so we will implement the procedure for the first 100 blocks. It is obvious that the script can be implemented and for the whole number of blocks (38351) , we are just using a partial solution to present our findings in the project. In figure 6 we can observe the outcome of the edge pair computation with their weights.

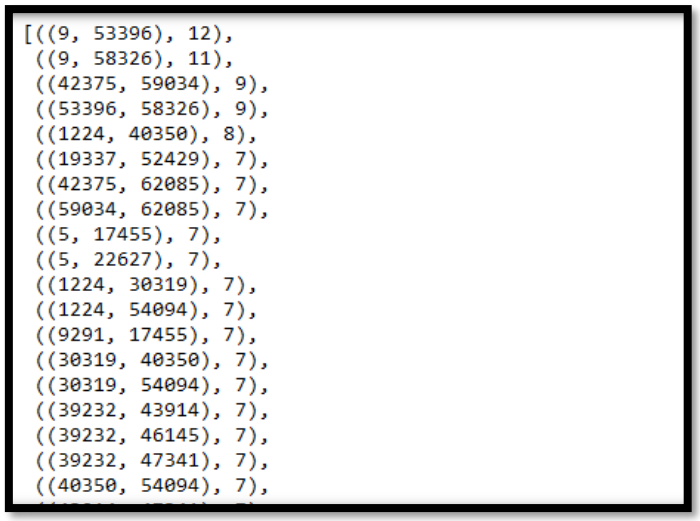

Figure 6 : Edge Pairs with their weights

Observing the figure 6 we can notice that the edge that connects the nodes 9 and 53396 has weight 12 etc.

It is necessary to be mentioned that saying that an edge has weight of 12 means that the two nodes have 12 common tokens. The bigger the weight is the more common tokens the two nodes share.

The total number of edges that we have for the first 100 blocks is **93.409.307**.

Pruning the Graph

Having weight equal with 1 is completely useless because implies that the two nodes have one common token meaning that they are more likely to by unmatching.

So, in this step of the assignment, we prune the graph for the edges that have weight < 2. The final number of edges after the pruning process is **6.183.412**.

Calculating the number of comparisons

Using the Meta Blocking allow us to have much less comparisons to solve the duplicates.

At the end of the procedure, we have the final number of comparisons we need to make to resolve the duplicates. The number of comparisons in Meta Blocking that is needed to resolve the duplicates is : **8,699,415**.

D. Create a function that takes as input two entities and computes their ***Jaccard*** similarity based on the attribute ***title.*** **You are not requested to perform any actual comparisons using this function**

Solution

To compute the Jaccard similarity of two entities we created a function named “Jaccard”. Before the function, the user is asked to type two entity IDs. The function takes the 2 inputs from the user and finds the Titles that correspond to the specific IDs. After that, the function computes the Jaccard similarity of the two Titles and returns the outcome to the user. If the Jaccard Similarity score is close to 1 then the two titles are similar if the score is close to 0 that means that the titles are not similar.

Example of Jaccard Similarity function

```
Give the first entity's ID (from 1 to 66879) : 42167
Give the second entitys's ID (from 1 to 66879) :42329


The first title is :  nhhâoccurrenceâextraction image sequences road traffic
The second title is :  hierarchical path views model based fragmentation transportation road types

The similarity of the entities based on the attribute TITLE is :  0.077
```

Figure 7 : Jaccard function output

As we can see the two Titles have one word in common and based on their count of the unique words of them their similarity is close to zero meaning that they do not have high similarity.