Python assignment

The second assignment is related to a research problem in the area of automated data cleaning and scientometrics. For this assignment you have to develop a program (classes, variables, and methods) that disambiguates scientific references in (a subset of) a patent database, called Patstat (see https://www.epo.org). This program could be used to connect the Patstat database with other database like the Web of Science, databases with scientific publications, to study the relationship between technology and science in a quantitative way. Such studies are important for policy makers, because they want to be informed on how investments in education eventually translate to patents.

You have to work together with the same group as before. The programming environments for Python (Anaconda Spyder or VS Code) can be downloaded for free (see e.g., https://www.anaconda.com/distribution/)

The deadline for the assignment is Monday, November 6, 23:59h. You have to upload your solutions via the Canvas. Be sure to collect all your files in a single *.zip or *.rar file. Compression might be required in the upload, otherwise the data files are too large.

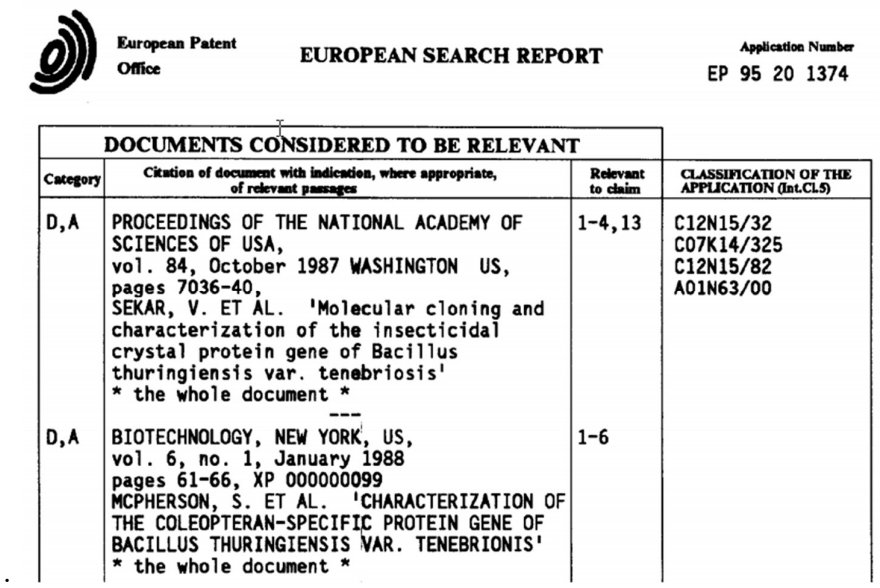
With your solutions you have to include a (brief) "Who did what?" section, explaining who did what for the completion of the assignment.

Have fun (again),

Emiel

Background

Scientometrics is a science that provides quantitative measures for evaluation of scientific output through analysis of bibliographic information. It is most commonly used in studies on impact and reach of scientific works. In economics its most prominent use can be found in policy evaluation and research on innovation. Such studies make use of the fact that bibliographic data is often linked to other economic phenomena. One example of such a relation are citations of scientific publications in patent publications. Figure 1 shows an excerpt from a patent publication.



European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 95 20 1374

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.5)
D,A	PROCEEDINGS OF THE NATIONAL ACADEMY OF SCIENCES OF USA, vol. 84, October 1987 WASHINGTON US, pages 7036-40, SEKAR, V. ET AL. 'Molecular cloning and characterization of the insecticidal crystal protein gene of Bacillus thuringiensis var. tenebriosis' * the whole document * ---	1-4,13	C12N15/32 C07K14/325 C12N15/82 A01N63/00
D,A	BIOTECHNOLOGY, NEW YORK, US, vol. 6, no. 1, January 1988 pages 61-66, XP 000000099 MCPHERSON, S. ET AL. 'CHARACTERIZATION OF THE COLEOPTERAN-SPECIFIC PROTEIN GENE OF BACILLUS THURINGIENSIS VAR. TENEBRIONIS' * the whole document *	1-6	

Fig 1. Scientific citation in a patent publication

Science, technology, and economy are intrinsically connected with each other. Understanding how undeniably constitutes valuable economic knowledge. However, in order to understand the connections between those phenomena a suitable methodological environment needs to be created.

PATSTAT is a product of the European Patent Office (EPO). It is a periodical snapshot of patent related information organized in a relational database model. Records on patent applications, their applicants and publications are available. A table with a code name "tls214_npl_publn" (often referred to as TLS214) stores information on bibliographic references like the one shown in Figure 1. This table contains more than 30 million records. The records, however, are often duplicated or inaccurate. Moreover, a full bibliographic reference is stored in only one attribute. This makes it problematic to query the table for relevant information, for example, to retrieve an author's name or the date of a specific publication.

Disambiguation in the context of data management refers to *the identification of unique entities within a dataset*. Such entities are identified by a unique identifier that can be assigned to many database records, which effectively describe the same bibliographic entity. The problem of data duplication and ambiguity arises due to (among other reasons):

a. Lack of consistent input (transcription) convention;
b. Variable level of input (detail) accuracy;

c. Missing data;
d. Different order of transcription of the same information;
e. Typos.

The following table excerpt illustrates the problem:

	npl_publn_id	npl_biblio
1	2219025	Codd, E. F., A Relational Model of Data for Large Shared Data Banks, Communications of the Association for Computing Machinery, Association for Computing ...
2	950805382	CODD, E.F.: A Relational Model of Data for Large Shared Data Banks. In: Comm. of the ACM, Vol. 13, Nr. 6, Juni 1970, S. 377-387
3	953756074	Codd, E.F., A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, 13(6):377-387 (1970).
4	955210884	E. Codd, A Relational Model of Data for Large Shared Data Banks, Communications of the ACM,vol. 13, No. 6, Jun. 1970, pp. 377-387.
5	955405309	Codd, E.F., A Relational Model of Data for Large Shared Data Banks, Jun. 1970, Communications of the ACM, vol. 13, No. 6, pp. 377-387.
6	955803441	Codd, E.F., A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, 13 (6):377-387 (1970).
7	956053490	Codd, A Relational Model of Data for Large Shared Data Banks, Communication of the ACM, vol. 13, No. 6, pp. 377-387, 1970.
8	956038611	Codd, E.F., A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, Jun. 1970, pp. 377-387, vol. 13, No. 6, Association for Com...
9	956911020	Codd, E.F., A relational Model of Data for Large Shared Data Banks, originally published in CACM, Jun. 1970,republished in Readings in Database Systems, 3rd ...
10	956866217	Codd, E.F., A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, 13 (6) : 377-387 (1970).
11	956880085	Codd, E.F. A Relational Model of Data for Large Shared Data Banks Communications of the ACM, vol. 13, No. 6, Jun. 1970, pp. 377-387.
12	957314767	Codd, E.F., A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, Jun. 1970, pp. 377-387, vol. 13, No. 6, Association for Com...
13	957626068	Codd, E. F., A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, Jun. 1970, pp. 377-387, vol. 13, No. 6.
14	957626152	Codd, E. F., A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, Jun. 1970, pp. 377-387, vol. 13, No. 6.
15	957626175	Codd, E. F., A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, Jun. 1970, pp. 377-387, vol. 13, No. 6.
16	957626239	Codd, E. F., A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, Jun. 1970, pp. 377-387, vol. 13, No. 6.
17	957626308	Codd, E. F., A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, Jun. 1970, pp. 377-387, vol. 13, No. 6.
18	957626375	Codd, E. F., A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, Jun. 1970, pp. 377-387, vol. 13, No. 6.

Fig 2. Example of 18 out of 56 records found by a simple search on the exact title match. Thus, even more records referring to the same entity may exist in the database.

All of the records shown above refer to the same entity – the paper by E.F. Codd on the relational database model (Codd, 1970). However, the references to the same entity are given in different ways or are simply duplicated. For example, record 7 does not contain Codd's name initial or month information, while record 8 contains full transcription of the abbreviated name "ACM" at the end of the string. All the records describe the same bibliographic entity but are treated as distinct entities by the primary key of the TLS214 relation – the "npl_publn_id" attribute.

Such a design makes it very difficult to use information in the table in a correct way. For example, say a researcher is interested in the relation between science and technology. She assesses that a scientific discovery is well proxied by a publication of a scientific paper, while a piece of technology can be modelled as a patent publication. However, due to the unsupervised procedure in which citations are added to the PATSTAT database it is difficult for her to specify a query that takes into account all the possible variation in records that describe the same bibliographic entity. As a result, the researcher is unable to properly count all of the scientific references to the same bibliographic entity. This results in incorrect patent statistics. As much as it may be possible to identify a single researcher and his body of work (like E.F. Codd), studies on a population of researchers are impossible without a prior cleansing and de-duplication of the PATSTAT database.

Research problem

The goal of this research is to *develop an automated method for the disambiguation of scientific references* of the original "tls214_npl_publn" table of the PATSTAT database. "Scientific references" refer to the types of records in the table that describe entities that can be classified as publications. Notice that not all records in the table are scientific references. Your method should specifically focus on providing 'the best possible results' for scientific references. The final result of the procedure is a table with *clusters of name variants* (records)

for each, unique scientific entity. The next table presents an example cluster with label 231 for the paper by E.F. Codd.

cluster_id	npl_publn_id	npl_biblio
231	2219025	Codd, E. F., A Relational Model of Data for Large Shared Data Banks, Communications of the Association for Computing Machinery, Association for Computing Machinery 13: Jun. 6, 1970, pp. 377-387, XP002219025.
231	950805382	CODD, E.F.: A Relational Model of Data for Large Shared Data Banks. In: Comm. of the ACM, Vol. 13, Nr. 6, Juni 1970, S. 377-387
231	953756074	Codd, E.F., A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, 13(6):377-387 (1970).
231	955210884	E. Codd, A Relational Model of Data for Large Shared Data Banks, Communications of the ACM,vol. 13, No. 6, Jun. 1970, pp. 377-387.
231	955405309	Codd, E.F., A Relational Model of Data for Large Shared Data Banks, Jun. 1970, Communications of the ACM, vol. 13, No. 6, pp. 377-387.
…	…	…
…	…	…

As an intermediate product you have to *extract* relevant bibliographic features from the string 'npl_biblio'. For example, from the string "*Codd, E. F., A Relational Model of Data for Large Shared Data Banks, Communications of the Association for Computing Machinery, Association for Computing Machinery 13: Jun. 6, 1970, pp. 377-387, XP002219025"*. You can extract the following bibliographic items:

- Author names;
- Paper title;
- Journal name;
- Volume*;
- Issue*;
- Edition;
- Pages (start and end)*;
- Publication year*;
- Publication month (date)*;
- ISSN and ISBN*;
- DOI*;
- XP number (a unique number used by Patstat sometimes) *;
- …

The items marked with a star have to be extracted, the other elements are optional, but useful. In general, it is often not possible to create 'perfect extractors', because the data is very heterogeneous. Your extractors should work in, say 90% to 95% of the cases. The extracted bibliographic items have to be stored in a multi-dimensional arrays or lists, which could be exported later to a table in SQL Server.

In general, you have to deliver *four end-products*:

1. A table with the *extracted bibliographic items* from the raw data.
2. A table with *clusters of name variants* (records) for each, unique scientific entity.
3. A table showing the performance of your clustering method with *precision-recall-f1 analysis.*
4. Based on the table in 3., you have to create plots for precision, recall, and f1. In the plots the cluster id's are on the horizontal axis and the performance values (ordered from high to low) are on the vertical axis.
    a. In addition, you have to report the average precision, recall, and f1 values, you have obtained on the whole set of clusters.
    b. Finally, show one (large) cluster with really good performance results, and show a cluster with poor performance.
    c. Note. For 4. you can put your results in an overview document.

All tables and plots should be directly available for inspection, without any work from my side.

In Python the data can be best represented in an array (matrix) or list (of lists) or as a data frame. After the processing you store the results as a table in MS SQL Server.

- Your programs first have to be designed for the table 'patstat_golden_set' (or the similar table patstat without a pre-classification), which contains ~15,000 records. For this solution you can maximal earn 8 out of 10 points.
- Finally, the clusters you have produced need to be evaluated on the golden set (i.e. the benchmark of a given classification) with precision-recall-f1 analysis (for this part you can earn 2 out of 10 points). Ideally, we want to have clusters with a high average value for the f1 performance metric, so scoring well on both precision and recall.

Algorithmic approach

You are free to come up with an algorithmic design for the described task. The only requirement is that you have to use regular expressions for the extraction and that your programs are also applicable to the general case of a larger database containing millions of records.

To give you some hints, the following algorithmic approach could be followed (as a guideline):

1. Data pre-processing (to harmonize the data).
    a. Make a connection with the SQL Server database, and collect the data from the table *patstat_golden_set*.
    b. Capture the data in the patstat_golden_set table as an array (multi-dimensional) or list or data frame and to try to reduce the variability of the data.
    c. Pre-clean the data by removing leading and trailing spaces.
    d. Remove multiple white spaces.
    e. Remove diacritics.
    f. Remove capitalization (if you think this is useful).
    g. Remove (irrelevant) interpunction.
    h. Etc.
2. Extract bibliographic items from scientific references.
    a. Use *regular expressions* to extract bibliographic information, based on labels, formats and ordering. As a pre-step you could harmonize certain data elements to a general format, for example, set 'page(s)' or 'seite(n)' (in German) to 'pages'. See the list of bibliographic elements on page 4 for an overview.
    b. You can also extract all numeric or alphabet information from the string for later use. Also, determine the length of the cleaned 'npl_biblio' string for later use.
    c. Store the extracted items in a table '*extracted_bib_items*'. The table can be created in SQL Server and inserted with records in the program. The table should be made unique by the column 'id', do not repeat the column 'npl_biblio', this will increase the size of table too much.
3. Make rules and clusters to identify duplicates.
    a. In this step, you have to determine the name variants and put them into unique clusters.
    b. A possible approach is to create *pairs of records* that are likely to be name variants based on rules. Those records are in some aspect(s) alike. Pairs are found by comparing records to each other according to some similarity measure defined by the rule. The most basic similarity measure is a Boolean value comparison of one of the evaluated labels. For example, all pairs of records that match on their ISBN, pages_start, pages_end, volume, issue, d_year, and d_month, attributes may be considered as a useful pair.
    c. It is a good idea to make rules that use some measure of string similarity. To computer the string similarity between a pair of records it is very useful to compute the *Levenshtein distance* (see

https://en.wikipedia.org/wiki/Levenshtein_distance) or alternatively the *Jaccard index* (see https://en.wikipedia.org/wiki/Jaccard_index), or some other measure. For example, if the string similarity is above some threshold, say 0.90 or so, you consider it to be a name variant. Be careful, with the use of numbers. Notice, that it is *not tractable* to compute the string similarity for all possible pairs in the database. However, string similarity is very useful to use in combination with a combination of bibliographic items, for example, a pair that matches on publication year, publication month, and is above 0.90 threshold value of string similarity.
Note. The computational efficiency of Jaccard and Levenshtein distance is $O(n^2)$. A more efficient alternative for theses distance metrics might be *tf-idf* (see https://en.wikipedia.org/wiki/Tf%E2%80%93idf), however it is more difficult to implement. Python offers online some libraries with an implementation of tf-idf together with pandas and scipy.

d. You could make a number of simple rules to detect similarity between pairs and give some (initial) score values to the rules. Later you could add up the scores for the pairs and consider pairs for clustering which are *above some threshold value*, i.e., the combination of rules could function as a distance measure. The scores can be collected in an adjacency matrix.
e. In the final step, you can connect the pairs (above the threshold) to each other to form a (*maximal*) *clique* https://en.wikipedia.org/wiki/Clique_(graph_theory), *community structure* or cluster. This can be done by several algorithms, for example,
    a. by finding all the *maximal cliques* between the pairs (see https://en.wikipedia.org/wiki/Clique_problem).
    b. by finding the *connected components* between the pairs (see https://en.wikipedia.org/wiki/Connected_component_(graph_theory))
    c. Another clustering method of your choice.

4. Data post-processing
    a. In the post-processing, records for which no duplicates are detected can be assigned to new, single record clusters.
    b. Finally, you can store your results in a table with the name '*patstat_clusters'*, with a unique cluster number and the original id's. The table is exported to SQL Server for easy inspection.
5. Validation
    a. You have to validate the performance of your clusters by benchmarking the results of your clustering method with the (perfect) clusters in the golden set with *precision-recall-f1 analysis* (see https://en.wikipedia.org/wiki/Precision_and_recall). For this part you have to implement methods to compute precision, recall, and f1.
    b. The precision-recall-f1 values can be used to optimize the rules, weights and algorithms you have created in the previous steps. *You want to obtain a high average value for the f1 performance*. Include simple plots (precision, recall, f1) and tables of the performance of your clusters and method.