Task 3.1

For this task, use the publications dataset. The dataset has been downloaded from

In [155…

```
import pandas as pd
import numpy as np


df_pub= pd.read_csv("Data/Task3/publications.csv")
df_pub
```

Out[155…

	id	title	authors	venue	year
0	304586	The WASA2 object-oriented workflow management ...	Gottfried Vossen, Mathias Weske	International Conference on Management of Data	1999
1	304587	A user-centered interface for querying distrib...	Isabel F. Cruz, Kimberly M. James	International Conference on Management of Data	1999
2	304589	World Wide Database-integrating the Web, CORBA...	Athman Bouguettaya, Boualem Benatallah, Lily H...	International Conference on Management of Data	1999
3	304590	XML-based information mediation with MIX	Chaitan Baru, Amarnath Gupta, Bertram Lud&#228...	International Conference on Management of Data	1999
4	304582	The CCUBE constraint object-oriented database ...	Alexander Brodsky, Victor E. Segal, Jia Chen, ...	International Conference on Management of Data	1999
...	...	...	...	...	...
995	conf/vldb/RamakrishnanR96	Modeling Design Versions	D. Janaki Ram, R. Ramakrishnan	VLDB	1996
996	conf/sigmod/BerchtoldK98	High-Dimensional Index Structures, Database Su...	Daniel A. Keim, Stefan Berchtold	SIGMOD Conference	1998
997	conf/sigmod/ChoALS03	LockX: A System for Efficiently Querying Secur...	SungRan Cho, Laks V. S. Lakshmanan, Divesh Sri...	SIGMOD Conference	2003
998	journals/sigmod/Winslett02a	David DeWitt Speaks Out	Marianne Winslett	SIGMOD Record	2002
999	conf/sigmod/AndersonAF98	Oracle Rdb's Record Caching Model	Richard Frank, Gopalan Arun, Richard Anderson	SIGMOD Conference	1998

1000 rows × 5 columns

Question 1. Perform pairwise comparison between the records in the dataset (publications.csv) to detect the duplicate records. To compare two records, follow the steps:

a. Ignore the pub_id.

In [156…

```
df_pu = df_pub.drop("id",axis=1)
df_pu
```

Out[156…

	title	authors	venue	year
0	The WASA2 object-oriented workflow management ...	Gottfried Vossen, Mathias Weske	International Conference on Management of Data	1999
1	A user-centered interface for querying distrib...	Isabel F. Cruz, Kimberly M. James	International Conference on Management of Data	1999
2	World Wide Database-integrating the Web, CORBA...	Athman Bouguettaya, Boualem Benatallah, Lily H...	International Conference on Management of Data	1999
3	XML-based information mediation with MIX	Chaitan Baru, Amarnath Gupta, Bertram Lud&#228...	International Conference on Management of Data	1999
4	The CCUBE constraint object-oriented database ...	Alexander Brodsky, Victor E. Segal, Jia Chen, ...	International Conference on Management of Data	1999
...	...	...	...	...
995	Modeling Design Versions	D. Janaki Ram, R. Ramakrishnan	VLDB	1996
996	High-Dimensional Index Structures, Database Su...	Daniel A. Keim, Stefan Berchtold	SIGMOD Conference	1998
997	LockX: A System for Efficiently Querying Secur...	SungRan Cho, Laks V. S. Lakshmanan, Divesh Sri...	SIGMOD Conference	2003
998	David DeWitt Speaks Out	Marianne Winslett	SIGMOD Record	2002
999	Oracle Rdb's Record Caching Model	Richard Frank, Gopalan Arun, Richard Anderson	SIGMOD Conference	1998

1000 rows × 4 columns

question 2 Use Levenshtein similarity for comparing the titles and computing the score (s_t)

In [157…

```python
import pandas
from py_stringmatching import similarity_measure as sm
df = df_pub.copy()

#adding a fake column to do a merge
df["test"] = 1

#Adding the row number column to dont loose in which row the data was
df["row"] = df.index

#merging to make pairs
rela = df.merge(df, on= "test", how = 'left', indicator = True).drop_duplicates()

#dropping similar rows data (comparing between themselfs)
rela = rela[rela.row_x != rela.row_y]
rela



#Dropping rows that will be compared twice E.g  title A compared to title B = Title B 
compared to Title A
group = rela[['row_x', 'row_y']].agg(frozenset, axis=1)
rela = (rela
 .groupby(group, as_index=False)  # you can also group by [group, 'Score']
 .agg(**{c: (c, 'first') for c in rela},
      )
)


#We reduced from 1Million to 495k pairs to compare
rela
```

Out[157…

	id_x	title_x	authors_x	venue_x	year_x	test	row_x	
0	304586	The WASA2 object-oriented workflow management ...	Gottfried Vossen, Mathias Weske	International Conference on Management of Data	1999	1	0	
1	335428	Adaptive multi-stage distance join processing	Hyoseop Shin, Bongki Moon, Sukho Lee	International Conference on Management of Data	2000	1	422	journa
2	335428	Adaptive multi-stage distance join processing	Hyoseop Shin, Bongki Moon, Sukho Lee	International Conference on Management of Data	2000	1	422	
3	335428	Adaptive multi-stage distance join processing	Hyoseop Shin, Bongki Moon, Sukho Lee	International Conference on Management of Data	2000	1	422	
4	335428	Adaptive multi-stage distance join processing	Hyoseop Shin, Bongki Moon, Sukho Lee	International Conference on Management of Data	2000	1	422	
...	...	...	...	...	...	...	...	
499495	276347	Replication, consistency, and practicality: ar...	Todd Anderson, Yuri Breitbart, Henry F. Korth,...	International Conference on Management of Data	1998	1	183	journa
499496	276347	Replication, consistency, and practicality: ar...	Todd Anderson, Yuri Breitbart, Henry F. Korth,...	International Conference on Management of Data	1998	1	183	journal
499497	276347	Replication, consistency, and practicality: ar...	Todd Anderson, Yuri Breitbart, Henry F. Korth,...	International Conference on Management of Data	1998	1	183	conf/sigmod/Ab

	id_x	title_x	authors_x	venue_x	year_x	test	row_x	
499498	276347	Replication, consistency, and practicality: ar...	Todd Anderson, Yuri Breitbart, Henry F. Korth,...	International Conference on Management of Data	1998	1	183	journals/s
499499	journals/sigmod/Winslett02a	David DeWitt Speaks Out	Marianne Winslett	SIGMOD Record	2002	1	998	

499500 rows × 14 columns

In [158…

```
rel= rela.copy()
rel = rel[["title_x","title_y"]]

lev_sim = sm.levenshtein.Levenshtein()



rel["s_t"]  = rel.apply(lambda x: lev_sim.get_sim_score(x['title_x'],  x['title_y']),
axis=1)
rel1 = rel.copy()
```

In [159…

```
rel1
```

Out[159…

	title_x	title_y	s_t
0	The WASA2 object-oriented workflow management ...	A user-centered interface for querying distrib...	0.211268
1	Adaptive multi-stage distance join processing	An Introduction to Deductive Database Language...	0.220339
2	Adaptive multi-stage distance join processing	Thémis: A Database Programming Language Handli...	0.228571
3	Adaptive multi-stage distance join processing	Integrating Information for On Demand Computing	0.212766
4	Adaptive multi-stage distance join processing	Enterprise Information Architectures -- They'r...	0.234375
...	...	...	...
499495	Replication, consistency, and practicality: ar...	WaveCluster: A Wavelet Based Clustering Approa...	0.191011
499496	Replication, consistency, and practicality: ar...	Priority Assignment in Real-Time Active Databases	0.260274
499497	Replication, consistency, and practicality: ar...	Aurora: A Data Stream Management System	0.191781
499498	Replication, consistency, and practicality: ar...	Database Research at UT Arlington	0.178082
499499	David DeWitt Speaks Out	Oracle Rdb's Record Caching Model	0.181818

499500 rows × 3 columns

question 3)Use Jaro similarity to compare the values in the authors field and compute (s_a)

In [160…
```
from py_stringmatching import similarity_measure as sm
df = df_pu.copy()
df["test"] = 1
rel= rela.copy()
rel = rel[["authors_x","authors_y"]]
jaro_sim = sm.jaro.Jaro()



rel["s_a"]  = rel.apply(lambda x: jaro_sim.get_raw_score(x['authors_x'],
x['authors_y']), axis=1)
rel2 = rel.copy()
rel2
```

Out[160…

	authors_x	authors_y	s_a
0	Gottfried Vossen, Mathias Weske	Isabel F. Cruz, Kimberly M. James	0.450585
1	Hyoseop Shin, Bongki Moon, Sukho Lee	Kotagiri Ramamohanarao, James Harland	0.446232
2	Hyoseop Shin, Bongki Moon, Sukho Lee	Anne Doucet, Véronique Benzaken	0.507691
3	Hyoseop Shin, Bongki Moon, Sukho Lee	Nelson Mendonça Mattos	0.510943
4	Hyoseop Shin, Bongki Moon, Sukho Lee	Wesley P. Melling	0.482026
...	...	...	...
499495	Todd Anderson, Yuri Breitbart, Henry F. Korth,...	Surojit Chatterjee, Gholamhosein Sheikholeslam...	0.564157
499496	Todd Anderson, Yuri Breitbart, Henry F. Korth,...	Rajendran M. Sivasankaran, Bhaskar Purimetla, ...	0.589344
499497	Todd Anderson, Yuri Breitbart, Henry F. Korth,...	Nesime Tatbul, Daniel J. Abadi, C. Erwin, Anur...	0.537256
499498	Todd Anderson, Yuri Breitbart, Henry F. Korth,...	Sharma Chakravarthy, Y. Alp Aslandogan, Ramez ...	0.576597
499499	Marianne Winslett	Richard Frank, Gopalan Arun, Richard Anderson	0.470152

499500 rows × 3 columns

question 4) Use the modified affine similarity for the conference (conf) attribute (s_c)

In [161…
```
from py_stringmatching import similarity_measure as sm
df = df_pu.copy()
df["test"] = 1



rel= rela.copy()
rel = rel[["venue_x","venue_y"]]

aff = sm.affine.Affine(gap_start = 1, gap_continuation = 0.1, \
                       sim_func = lambda s1, s2: (int(1 if s1 == s2 else 0)))



rel["s_c"]  = rel.apply(lambda x: aff.get_raw_score(x['venue_x'],  x['venue_y'])/
min(len(x['venue_x']),len(x['venue_y'])), axis=1)
rel3 = rel.copy()
```

In [162…

```
rel3
```

Out[162…

	venue_x	venue_y	s_c
0	International Conference on Management of Data	International Conference on Management of Data	1.000000
1	International Conference on Management of Data	VLDB J.	-0.671428
2	International Conference on Management of Data	VLDB J.	-0.671428
3	International Conference on Management of Data	VLDB	-1.249999
4	International Conference on Management of Data	SIGMOD Conference	0.370588
...	...	...	...
499495	International Conference on Management of Data	VLDB J.	-0.671428
499496	International Conference on Management of Data	VLDB J.	-0.671428
499497	International Conference on Management of Data	SIGMOD Conference	0.370588
499498	International Conference on Management of Data	SIGMOD Record	-0.230769
499499	SIGMOD Record	SIGMOD Conference	0.446154

499500 rows × 3 columns

question5) Use Match (1) / Mismatch (0) for the year (s_y)

In [163…

```
df = df_pu.copy()
dic = {True:1, False:0}
df["test"] = 1



rel= rela.copy()
rel = rel[["year_x","year_y"]]
rel["s_y"] = rel["year_x"] ==rel["year_y"]
rel["s_y"] = rel["s_y"].replace(dic)
rel4 = rel.copy()
```

question6)Use the formula rec_sim= 0.5*s_t + 0.2*s_a +0.2*s_c +0.1* s_y to combine the scores and compute the final score.

In [164…

```
df = rela.copy()
df["rec_sim"] = (rel1["s_t"]*.5)+ (rel2["s_a"]*.2) +  (rel3["s_c"]*.2)  +
(rel4["s_y"]*.1)
df
```

Out[164…

	id_x	title_x	authors_x	venue_x	year_x	test	row_x	
0	304586	The WASA2 object-oriented workflow management ...	Gottfried Vossen, Mathias Weske	International Conference on Management of Data	1999	1	0	
1	335428	Adaptive multi-stage distance join processing	Hyoseop Shin, Bongki Moon, Sukho Lee	International Conference on Management of Data	2000	1	422	journa
2	335428	Adaptive multi-stage distance join processing	Hyoseop Shin, Bongki Moon, Sukho Lee	International Conference on Management of Data	2000	1	422	
3	335428	Adaptive multi-stage distance join processing	Hyoseop Shin, Bongki Moon, Sukho Lee	International Conference on Management of Data	2000	1	422	
4	335428	Adaptive multi-stage distance join processing	Hyoseop Shin, Bongki Moon, Sukho Lee	International Conference on Management of Data	2000	1	422	
...	...	...	...	...	...	...	...	
499495	276347	Replication, consistency, and practicality: ar...	Todd Anderson, Yuri Breitbart, Henry F. Korth,...	International Conference on Management of Data	1998	1	183	journa
499496	276347	Replication, consistency, and practicality: ar...	Todd Anderson, Yuri Breitbart, Henry F. Korth,...	International Conference on Management of Data	1998	1	183	journal
499497	276347	Replication, consistency, and practicality: ar...	Todd Anderson, Yuri Breitbart, Henry F. Korth,...	International Conference on Management of Data	1998	1	183	conf/sigmod/Ab
499498	276347	Replication, consistency, and practicality: ar...	Todd Anderson, Yuri Breitbart, Henry F. Korth,...	International Conference on Management of Data	1998	1	183	journals/s
499499	journals/sigmod/Winslett02a	David DeWitt Speaks Out	Marianne Winslett	SIGMOD Record	2002	1	998	

499500 rows × 15 columns

question 7) Report the records with rec_sim > 0.7 as duplicate records by storing the ids of both records in a list.

In [165…

```
df_dup = df.loc[df["rec_sim"]> 0.7]

df_dup.head(50)

df_du = df_dup[["id_x","id_y"]]
df_du.reset_index(inplace=True,drop=True)

#DF of all records (with repetition of some ids)
df_du
```

Out[165…

	id_x	id_y
0	335429	conf/sigmod/ChoSG00
1	335465	conf/sigmod/Weininger00
2	336573	conf/sigmod/HsuLG00
3	336560	conf/sigmod/ChenDLT00
4	336587	conf/sigmod/BressanGOT00
...	...	...
148	276353	conf/sigmod/BerchtoldK98
149	276355	conf/sigmod/WhiteCF98
150	276345	conf/sigmod/LometW98
151	276341	conf/sigmod/FernandezFKLS98
152	276347	conf/sigmod/AndersonBKW98

153 rows × 2 columns

In [166…

```
#Storing in a dictionay to eliminate some repeated ids
dic
dic ={}

for i in range(len(df_du["id_x"])):
    key = df_du.loc[i,"id_x"]

    if key not in dic:
        dic[key] = []
        dic[key].append(df_du.loc[i,"id_y"])

    else:
        dic[key].append(df_du.loc[i,"id_y"])

print("total ids repeated",len(dic.keys()))
dic1 = dic.copy()
```

```
total ids repeated 116
```

QUESTION 8)In the table pub_mappings.csv, you can find the actual mappings (the ids of the correct duplicate records). Compare the accuracy of this method by counting the number of duplicate records that you discovered correctly.

In [167…

```
dic =  dic1.copy()
real = pd.read_csv("Data/Task3/pub_mappings.csv")

#Real count
real_len = len(real)
#Estimation count
estimation = len(dic.keys())
error = (abs(real_len - estimation)  / real_len) * 100
error

#Precision in terms of counts number
precision = 100 -error
print("precision in count is ",precision,"%")



#Adjustinf real df columns names
real.columns = ['idDBLP', 'idACM']

#Getting just the first element of dic lists to pair
for i in dic.keys():
    dic[i]=  dic[i][0]

#creating a df with dictionary elements (estimated elements)
pred = pd.DataFrame(dic.keys(),dic.values())
pred = pred.reset_index()
pred.columns = ['idDBLP', 'idACM']
real['idACM']=real['idACM'].astype(str)



#getting coincidence between real and estimated dfs
df_merged_1 = pred.merge(real, how='outer', on=['idDBLP', 'idACM'], indicator=True)

#Present in both lists
a = df_merged_1.loc[df_merged_1["_merge"] == "both"]

print("Found ",len(a)," coincidences Out of", len(real))
```

```
precision in count is  75.26881720430107 %
Found  89  coincidences Out of 93
```

question 9) Record the running time of the method when processing the pairwise similarity between the 1000 records.

In [168…

```
import time
start_time = time.time()
```

```
import pandas
from py_stringmatching import similarity_measure as sm

df = df_pub.copy()
df["test"] = 1
df["row"] = df.index

rela = df.merge(df, on= "test", how = 'left', indicator = True).drop_duplicates()
rela = rela[rela.row_x != rela.row_y]
rela

group = rela[['row_x', 'row_y']].agg(frozenset, axis=1)
rela = (rela
 .groupby(group, as_index=False)  # you can also group by [group, 'Score']
 .agg(**{c: (c, 'first') for c in rela},
     )
)
rela

rel= rela.copy()
rel = rel[["title_x","title_y"]]

lev_sim = sm.levenshtein.Levenshtein()


rel["s_t"]  = rel.apply(lambda x: lev_sim.get_sim_score(x['title_x'],  x['title_y']),
axis=1)
rel1 = rel.copy()

from py_stringmatching import similarity_measure as sm

rel= rela.copy()
rel = rel[["authors_x","authors_y"]]
jaro_sim = sm.jaro.Jaro()


rel["s_a"]  = rel.apply(lambda x: jaro_sim.get_raw_score(x['authors_x'],
x['authors_y']), axis=1)
rel2 = rel.copy()
rel2

from py_stringmatching import similarity_measure as sm



rel= rela.copy()
rel = rel[["venue_x","venue_y"]]
```

```
aff = sm.affine.Affine(gap_start = 1, gap_continuation = 0.1, \
                       sim_func = lambda s1, s2: (int(1 if s1 == s2 else 0)))


rel["s_c"]  = rel.apply(lambda x: aff.get_raw_score(x['venue_x'],  x['venue_y'])/
min(len(x['venue_x']),len(x['venue_y'])), axis=1)
rel3 = rel.copy()


dic = {True:1, False:0}
df["test"] = 1


rel= rela.copy()
rel = rel[["year_x","year_y"]]
rel["s_y"] = rel["year_x"] ==rel["year_y"]
rel["s_y"] = rel["s_y"].replace(dic)
rel4 = rel.copy()

df = rela.copy()
df["rec_sim"] = (rel1["s_t"]*.5)+ (rel2["s_a"]*.2) +  (rel3["s_c"]*.2)  +
(rel4["s_y"]*.1)
df

df_dup = df.loc[df["rec_sim"]> 0.7]

df_dup.head(50)

df_du = df_dup[["id_x","id_y"]]
df_du.reset_index(inplace=True,drop=True)


df_du

#Storing in a dictionay
dic
dic ={}

for i in range(len(df_du["id_x"])):
    key = df_du.loc[i,"id_x"]

    if key not in dic:
        dic[key] = []
        dic[key].append(df_du.loc[i,"id_y"])

    else:
        dic[key].append(df_du.loc[i,"id_y"])
```

```
dic
dic1 = dic.copy()
print("total ids repeated",len(dic1.keys()))



#saving total time in a variable
q1 = "--- %s seconds ---" % (time.time() - start_time)
print(q1)
```

```
total ids repeated 116
--- 398.79845905303955 seconds ---
```

Task 3.2

Repeat question1 but compare only the records from table publications_B1.csv with those in publications_B2.csv (do not compare the records that exist in the same file). That is, you will compare each record from the 500 records in the first table with all records in the second table. Compute the accuracy and the running time and compare the running time with the running time that was obtained in question 1.

In [169…

```
import time
start_time = time.time()

import pandas
from py_stringmatching import similarity_measure as sm
df_pub1= pd.read_csv("Data/Task3/publications_B1.csv")
df_pub2= pd.read_csv("Data/Task3/publications_B2.csv")
df1 = df_pub1.copy()
df2 = df_pub2.copy()
df1["test"] = 1
df1["row"] = df1.index

df2["test"] = 1
df2["row"] = df2.index

rela = df1.merge(df2, on= "test", how = 'left', indicator = True).drop_duplicates()



rel= rela.copy()
rel = rel[["title_x","title_y"]]

lev_sim = sm.levenshtein.Levenshtein()



rel["s_t"]  = rel.apply(lambda x: lev_sim.get_sim_score(x['title_x'],  x['title_y']),
axis=1)
rel1 = rel.copy()

from py_stringmatching import similarity_measure as sm
```

```
rel= rela.copy()
rel = rel[["authors_x","authors_y"]]
jaro_sim = sm.jaro.Jaro()


rel["s_a"]  = rel.apply(lambda x: jaro_sim.get_raw_score(x['authors_x'],
x['authors_y']), axis=1)
rel2 = rel.copy()
rel2







rel= rela.copy()
rel = rel[["venue_x","venue_y"]]

aff = sm.affine.Affine(gap_start = 1, gap_continuation = 0.1, \
                       sim_func = lambda s1, s2: (int(1 if s1 == s2 else 0)))


rel["s_c"]  = rel.apply(lambda x: aff.get_raw_score(x['venue_x'],  x['venue_y'])/
min(len(x['venue_x']),len(x['venue_y'])), axis=1)
rel3 = rel.copy()



dic = {True:1, False:0}



rel= rela.copy()
rel = rel[["year_x","year_y"]]
rel["s_y"] = rel["year_x"] ==rel["year_y"]
rel["s_y"] = rel["s_y"].replace(dic)
rel4 = rel.copy()

df = rela.copy()
df["rec_sim"] = (rel1["s_t"]*.5)+ (rel2["s_a"]*.2) +  (rel3["s_c"]*.2)  +
(rel4["s_y"]*.1)
df

df_dup = df.loc[df["rec_sim"]> 0.7]

df_dup.head(50)

df_du = df_dup[["id_x","id_y"]]
```

```
df_du.reset_index(inplace=True,drop=True)




#Storing in a dictionay
dic
dic ={}

for i in range(len(df_du["id_x"])):
    key = df_du.loc[i,"id_x"]

    if key not in dic:
        dic[key] = []
        dic[key].append(df_du.loc[i,"id_y"])

    else:
        dic[key].append(df_du.loc[i,"id_y"])
dic
dic2 = dic.copy()

print("total ids repeated",len(dic2.keys()))



q2 = "--- %s seconds ---" % (time.time() - start_time)
print(q2)
df_du
```

```
total ids repeated 93
--- 167.38291811943054 seconds ---
```

Out[169…

	id_x	id_y
0	304589	conf/sigmod/BouguettayaBH99
1	304590	conf/sigmod/BaruGLMPVC99
2	306112	journals/sigmod/JenningsNF98
3	304573	conf/sigmod/BraumandlKK99
4	304568	conf/sigmod/JarkeQBLMS99
...	...	...
98	362136	journals/sigmod/Bussche00
99	363954	journals/tods/BaralisW00
100	373709	journals/sigmod/MeltonMJKSZ01
101	375664	conf/sigmod/HanPDW01
102	362091	journals/sigmod/KantM00

103 rows × 2 columns

In [170…

```
#comparing total times
print("total time q1=" , q1 )
print("total time q2=" , q2 )
```

```
total time q1= --- 398.79845905303955 seconds ---
total time q2= --- 167.38291811943054 seconds ---
```

In [171…

```
dic =  dic2.copy()
real = pd.read_csv("Data/Task3/pub_mappings.csv")

#Real count
real_len = len(real)
#Estimation count
estimation = len(dic.keys())
error = (abs(real_len - estimation)  / real_len) * 100
error

#Precision in terms of counts number
precision = 100 -error
print("precision in count is ",precision,"%")



#Adjustinf real df columns names
real.columns = ['idDBLP', 'idACM']

#Getting just the first element of dic lists to pair
for i in dic.keys():
    dic[i]=  dic[i][0]

#creating a df with dictionary elements (estimated elements)
pred = pd.DataFrame(dic.keys(),dic.values())
pred = pred.reset_index()
pred.columns = ['idDBLP', 'idACM']
pred['idACM']=pred['idACM'].astype(str)
real['idACM']=real['idACM'].astype(str)



#getting coincidence between real and estimated dfs
df_merged_1 = pred.merge(real, how='outer', on=['idDBLP', 'idACM'], indicator=True)

#Present in both lists
a = df_merged_1.loc[df_merged_1["_merge"] == "both"]

print("Found ",len(a)," coincidences Out of", len(real))


df_merged_1
```

```
precision in count is  100.0 %
Found  89  coincidences Out of 93
```

Out[171…

	idDBLP	idACM	_merge
0	conf/sigmod/BouguettayaBH99	304589	both
1	conf/sigmod/BaruGLMPVC99	304590	both
2	journals/sigmod/JenningsNF98	306112	both
3	conf/sigmod/BraumandlKK99	304573	both
4	conf/sigmod/JarkeQBLMS99	304568	both
...	...	...	...
92	journals/sigmod/KantM00	362091	both
93	conf/sigmod/Larson01	375792	right_only
94	conf/sigmod/ClossmanSHKPB98	276352	right_only
95	journals/sigmod/SnodgrassGIMSU98	290599	right_only
96	journals/sigmod/OukselS99	309849	right_only

97 rows × 3 columns

In [ ]: