

ds1 notebook output

December 5, 2025

0.1 Project

0.1.1 Constance Streitman (2253221)

```
[1]: import pandas as pd
from pathlib import Path

data_dir = Path(
    r"C:\Users\Constance\Documents\foar skuul frum kani\sem 3 - uh fall_
↳2025\data mining (ds1)\dblp-project"
)

json_files = sorted(data_dir.glob("dblp-ref-*.json"))
print(json_files)    # sanity check

dfs = []
for f in json_files:
    print("Loading", f.name)
    df_part = pd.read_json(f, lines=True)    # full file
    dfs.append(df_part)

dblp = pd.concat(dfs, ignore_index=True)

print(dblp.shape)
dblp.head()
```

```
[WindowsPath('C:/Users/Constance/Documents/foar skuul frum kani/sem 3 - uh fall
2025/data mining (ds1)/dblp-project/dblp-ref-0.json'),
WindowsPath('C:/Users/Constance/Documents/foar skuul frum kani/sem 3 - uh fall
2025/data mining (ds1)/dblp-project/dblp-ref-1.json'),
WindowsPath('C:/Users/Constance/Documents/foar skuul frum kani/sem 3 - uh fall
2025/data mining (ds1)/dblp-project/dblp-ref-2.json'),
WindowsPath('C:/Users/Constance/Documents/foar skuul frum kani/sem 3 - uh fall
2025/data mining (ds1)/dblp-project/dblp-ref-3.json')]
Loading dblp-ref-0.json
Loading dblp-ref-1.json
Loading dblp-ref-2.json
Loading dblp-ref-3.json
(3079007, 8)
```

```

[1]:                                     abstract \
0 The purpose of this study is to develop a lear...
1 This paper describes the design and implementa...
2 This article applied GARCH model instead AR or...
3                                     NaN
4                                     NaN

                                     authors  n_citation \
0 [Makoto Satoh, Ryo Muramatsu, Mizue Kayama, Ka...      0
1 [Gareth Beale, Graeme Earl]      50
2 [Altaf Hossain, Faisal Zaman, Mohammed Nasser,...    50
3 [Jea-Bum Park, Byungmok Kim, Jian Shen, Sun-Yo...      0
4 [Giovanna Guerrini, Isabella Merlo]      2

                                     references \
0 [51c7e02e-f5ed-431a-8cf5-f761f266d4be, 69b625b...
1 [10482dd3-4642-4193-842f-85f3b70fcf65, 3133714...
2 [2d84c0f2-e656-4ce7-b018-90eda1c132fe, a083a1b...
3 [8c78e4b0-632b-4293-b491-85b1976675e6, 9cdc54f...
4                                     NaN

                                     title \
0 Preliminary Design of a Network Protocol Learn...
1 A methodology for the physically accurate visu...
2 Comparison of GARCH, Neural Network and Suppor...
3 Development of Remote Monitoring and Control D...
4 Reasonig about Set-Oriented Methods in Object ...

                                     venue  year \
0 international conference on human-computer int... 2013
1 visual analytics science and technology 2011
2 pattern recognition and machine intelligence 2009
3                                     2011
4                                     1998

                                     id
0 00127ee2-cb05-48ce-bc49-9de556b93346
1 001c58d3-26ad-46b3-ab3a-c1e557d16821
2 001c8744-73c4-4b04-9364-22d31a10dbf1
3 00338203-9eb3-40c5-9f31-cbac73a519ec
4 0040b022-1472-4f70-a753-74832df65266

```

simplifying the dataset a little

```

[2]: needed_cols = [
    "id",          #str
    "title",       #str
    "authors",     #list of str

```

```

    "venue",      #str
    "year",       #int
    "n_citation", #int
    "references", #list of str
    "abstract"    #str
]
dblp = dblp[[c for c in needed_cols if c in dblp.columns]].copy()
dblp.head()

```

```

[2]:
                                id \
0  00127ee2-cb05-48ce-bc49-9de556b93346
1  001c58d3-26ad-46b3-ab3a-c1e557d16821
2  001c8744-73c4-4b04-9364-22d31a10dbf1
3  00338203-9eb3-40c5-9f31-cbac73a519ec
4  0040b022-1472-4f70-a753-74832df65266

                                title \
0  Preliminary Design of a Network Protocol Learn...
1  A methodology for the physically accurate visu...
2  Comparison of GARCH, Neural Network and Suppor...
3  Development of Remote Monitoring and Control D...
4  Reasonig about Set-Oriented Methods in Object ...

                                authors \
0  [Makoto Satoh, Ryo Muramatsu, Mizue Kayama, Ka...
1  [Gareth Beale, Graeme Earl]
2  [Altaf Hossain, Faisal Zaman, Mohammed Nasser,...
3  [Jea-Bum Park, Byungmok Kim, Jian Shen, Sun-Yo...
4  [Giovanna Guerrini, Isabella Merlo]

                                venue  year  n_citation \
0  international conference on human-computer int...  2013         0
1  visual analytics science and technology  2011        50
2  pattern recognition and machine intelligence  2009        50
3  2011                                         0
4  1998                                         2

                                references \
0  [51c7e02e-f5ed-431a-8cf5-f761f266d4be, 69b625b...
1  [10482dd3-4642-4193-842f-85f3b70fcf65, 3133714...
2  [2d84c0f2-e656-4ce7-b018-90eda1c132fe, a083a1b...
3  [8c78e4b0-632b-4293-b491-85b1976675e6, 9cdc54f...
4  NaN

                                abstract
0  The purpose of this study is to develop a lear...
1  This paper describes the design and implementa...

```

```

2 This article applied GARCH model instead AR or...
3                                     NaN
4                                     NaN

```

getting rid of some things with unnecessary NAs, cleaning up, whatever

```

[3]: dblp = dblp.dropna(subset=["id", "title"])

dblp["n_citation"] = pd.to_numeric(dblp["n_citation"], errors="coerce").
    ↪ fillna(0)
dblp["n_citation"] = dblp["n_citation"].astype("int32")

def fix_authors(a):
    if isinstance(a, list):
        return [str(x).strip() for x in a if str(x).strip()]
    if pd.isna(a):
        return []
    return [str(a).strip()]

dblp["authors"] = dblp["authors"].apply(fix_authors)

dblp["n_authors"] = dblp["authors"].apply(len)

def fix_refs(r):
    if isinstance(r, list):
        return [str(x).strip() for x in r if str(x).strip()]
    if pd.isna(r):
        return []
    return [str(r).strip()]

dblp["references"] = dblp["references"].apply(fix_refs)
dblp["n_references"] = dblp["references"].apply(len)

[4]: dblp["text"] = (dblp["title"] + " " + dblp["abstract"]).str.lower()
dblp.head()

```

```

[4]:
                                     id \
0  00127ee2-cb05-48ce-bc49-9de556b93346
1  001c58d3-26ad-46b3-ab3a-c1e557d16821
2  001c8744-73c4-4b04-9364-22d31a10dbf1
3  00338203-9eb3-40c5-9f31-cbac73a519ec
4  0040b022-1472-4f70-a753-74832df65266

                                     title \
0  Preliminary Design of a Network Protocol Learn...
1  A methodology for the physically accurate visu...
2  Comparison of GARCH, Neural Network and Suppor...
3  Development of Remote Monitoring and Control D...

```

4 Reasonig about Set-Oriented Methods in Object ...

	authors \
0	[Makoto Satoh, Ryo Muramatsu, Mizue Kayama, Ka...
1	[Gareth Beale, Graeme Earl]
2	[Altaf Hossain, Faisal Zaman, Mohammed Nasser,...
3	[Jea-Bum Park, Byungmok Kim, Jian Shen, Sun-Yo...
4	[Giovanna Guerrini, Isabella Merlo]

	venue	year	n_citation \
0	international conference on human-computer int...	2013	0
1	visual analytics science and technology	2011	50
2	pattern recognition and machine intelligence	2009	50
3		2011	0
4		1998	2

	references \
0	[51c7e02e-f5ed-431a-8cf5-f761f266d4be, 69b625b...
1	[10482dd3-4642-4193-842f-85f3b70fcf65, 3133714...
2	[2d84c0f2-e656-4ce7-b018-90eda1c132fe, a083a1b...
3	[8c78e4b0-632b-4293-b491-85b1976675e6, 9cdc54f...
4	[]

	abstract	n_authors	n_references \
0	The purpose of this study is to develop a lear...	8	2
1	This paper describes the design and implementa...	2	13
2	This article applied GARCH model instead AR or...	4	2
3	NaN	5	2
4	NaN	2	0

	text
0	preliminary design of a network protocol learn...
1	a methodology for the physically accurate visu...
2	comparison of garch, neural network and suppor...
3	NaN
4	NaN

```
[5]: print("Before:", dblp.shape)
      dblp = dblp.dropna(subset=["abstract", "text"]) #dropping everything with NaN
      print("After:", dblp.shape)
      dblp.head()
```

Before: (3079007, 11)

After: (2548532, 11)

```
[5]: id \
0    00127ee2-cb05-48ce-bc49-9de556b93346
1    001c58d3-26ad-46b3-ab3a-c1e557d16821
```

```

2  001c8744-73c4-4b04-9364-22d31a10dbf1
10 00a119c4-d367-4607-b3c8-b237f2971bff
12 00bcf2d5-1592-46b0-81fd-933f90b5ecca

```

```

                                title \
0  Preliminary Design of a Network Protocol Learn...
1  A methodology for the physically accurate visu...
2  Comparison of GARCH, Neural Network and Suppor...
10 Identifying Psychological Theme Words from Emo...
12 Multisymplectic Spectral Methods for the Gross...

```

```

                                authors \
0  [Makoto Satoh, Ryo Muramatsu, Mizue Kayama, Ka...
1                                [Gareth Beale, Graeme Earl]
2  [Altaf Hossain, Faisal Zaman, Mohammed Nasser,...
10 [Ankita Brahmachari, Priya Singh, Avdhesh Garg...
12 [Alvaro L. Islas, Constance M. Schober]

```

```

                                venue year n_citation \
0  international conference on human-computer int... 2013          0
1                                visual analytics science and technology 2011          50
2                                pattern recognition and machine intelligence 2009          50
10                               2013          0
12 international conference on conceptual structures 2002          50

```

```

                                references \
0  [51c7e02e-f5ed-431a-8cf5-f761f266d4be, 69b625b...
1  [10482dd3-4642-4193-842f-85f3b70fcf65, 3133714...
2  [2d84c0f2-e656-4ce7-b018-90eda1c132fe, a083a1b...
10 [84d47128-58d0-4187-aa44-389fde7d5c83, e0dce69...
12                                []

```

```

                                abstract n_authors \
0  The purpose of this study is to develop a lear...      8
1  This paper describes the design and implementa...      2
2  This article applied GARCH model instead AR or...      4
10 Recent achievements in Natural Language Proces...      4
12 Recently, Bridges and Reich introduced the con...      2

```

```

n_references                                text
0          2  preliminary design of a network protocol learn...
1          13 a methodology for the physically accurate visu...
2          2  comparison of garch, neural network and suppor...
10         3  identifying psychological theme words from emo...
12         0  multisymplectic spectral methods for the gross...

```

```
[6]: dblp = dblp.drop(columns=["id", "references"], errors="ignore")
      dblp.head()
```

```
[6]:
```

	title \
0	Preliminary Design of a Network Protocol Learn...
1	A methodology for the physically accurate visu...
2	Comparison of GARCH, Neural Network and Suppor...
10	Identifying Psychological Theme Words from Emo...
12	Multisymplectic Spectral Methods for the Gross...

	authors \
0	[Makoto Satoh, Ryo Muramatsu, Mizue Kayama, Ka...
1	[Gareth Beale, Graeme Earl]
2	[Altaf Hossain, Faisal Zaman, Mohammed Nasser,...
10	[Ankita Brahmachari, Priya Singh, Avdhesh Garg...
12	[Alvaro L. Islas, Constance M. Schober]

	venue	year	n_citation \
0	international conference on human-computer int...	2013	0
1	visual analytics science and technology	2011	50
2	pattern recognition and machine intelligence	2009	50
10		2013	0
12	international conference on conceptual structures	2002	50

	abstract	n_authors \
0	The purpose of this study is to develop a lear...	8
1	This paper describes the design and implementa...	2
2	This article applied GARCH model instead AR or...	4
10	Recent achievements in Natural Language Proces...	4
12	Recently, Bridges and Reich introduced the con...	2

	n_references	text
0	2	preliminary design of a network protocol learn...
1	13	a methodology for the physically accurate visu...
2	2	comparison of garch, neural network and suppor...
10	3	identifying psychological theme words from emo...
12	0	multisymplectic spectral methods for the gross...

1 TASK #1: EXPLORATORY DATA ANALYSIS

```
[7]: print("Total papers:", len(dblp))

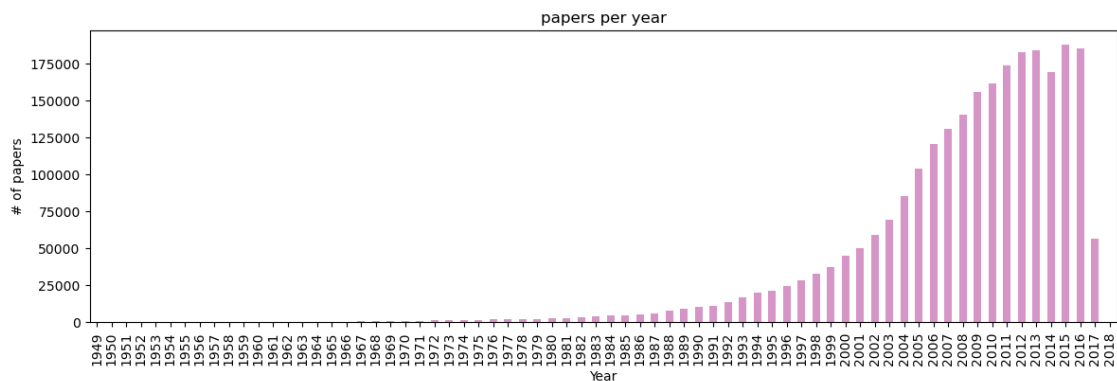
ppy = dblp.groupby("year").size().rename("paper_count")
ppy.head(), ppy.tail()
```

Total papers: 2548532

```
[7]: (year
      1949      1
      1950      2
      1951      2
      1952      7
      1953     21
      Name: paper_count, dtype: int64,
      year
      2014    169248
      2015    188143
      2016    185342
      2017     56552
      2018         4
      Name: paper_count, dtype: int64)
```

```
[8]: import matplotlib.pyplot as plt

      ppy.plot(kind="bar", figsize=(14,4), color="#d695c7")
      plt.xlabel("Year")
      plt.ylabel("# of papers")
      plt.title("papers per year")
      plt.show()
```



```
[9]: popvenues = (
      dblp.groupby("venue")["n_citation"]
      .agg(["count", "mean", "median"])
      .sort_values("count", ascending=False)
      )

      popvenues.head(20)
```

```
[9]:
```

venue	count	mean	median
-------	-------	------	--------

	311564	22.746258	3.0
Lecture Notes in Computer Science	30429	31.546189	13.0
international conference on acoustics, speech, ...	26086	30.365407	21.0
international conference on robotics and automa...	18562	52.867902	50.0
international conference on image processing	17417	31.219843	19.0
international conference on communications	16447	26.832857	11.0
international symposium on circuits and systems	16177	23.009891	9.0
global communications conference	15763	27.083994	13.0
intelligent robots and systems	14069	31.646599	21.0
international geoscience and remote sensing sym...	13914	13.491232	1.0
Applied Mathematics and Computation	12956	24.874035	9.0
vehicular technology conference	12145	22.109510	5.0
conference on decision and control	12119	23.792062	8.0
human factors in computing systems	11356	50.317101	39.0
IEEE Transactions on Information Theory	10001	100.394061	50.0
IEEE Transactions on Signal Processing	9819	72.337713	50.0
computer vision and pattern recognition	9765	99.567537	50.0
Discrete Mathematics	9764	26.807558	14.0
European Journal of Operational Research	9692	53.063764	32.0
Neurocomputing	9352	25.613559	11.0

```
[ ]: from sklearn.feature_extraction.text import TfidfVectorizer

samplesubset = (dblp["text"].sample(200_000, random_state=2253221)) ##### used
↳ subset so it didnt take forever

tfidf = TfidfVectorizer(
    max_features=5000,
    stop_words="english",
    min_df=50,
    max_df = 0.4,
    ngram_range=(1, 2)
)
X = tfidf.fit_transform(samplesubset)

terms = tfidf.get_feature_names_out()
mean_tfidf = X.mean(axis=0).A1 #to make it percapita

term_scores = (
    pd.DataFrame({"term": terms, "score": mean_tfidf})
    .sort_values("score", ascending=False)
)

boiler_phrases = {
    "paper presents",
    "paper present",
    "paper proposes",
```

```

    "paper propose",
    "paper describes",
    "proposed method",
    "proposed approach",
    "proposed algorithm",
    "experimental results",
    "propose new",
    "state art",
    "real world",
    "case study",
    "ad hoc",
    "et al",
}

term_scores = term_scores[~term_scores["term"].isin(boiler_phrases)]

bigrams = term_scores[term_scores["term"].str.contains(" ")]
bigrams.head(30)

```

```

[ ]:
      term      score
3715  real time  0.004500
4198  simulation results  0.002596
2450  large scale  0.002336
2959  neural network  0.002335
4100  sensor networks  0.001949
2960  neural networks  0.001891
4961  wireless sensor  0.001698
1155  decision making  0.001596
2807  model based  0.001580
2063  high level  0.001456
2065  high performance  0.001435
1126  data sets  0.001426
2605  machine learning  0.001413
2741  method based  0.001405
4617  time series  0.001340
204   algorithm based  0.001308
3907  results demonstrate  0.001299
1941  genetic algorithm  0.001266
1120  data mining  0.001266
3911  results proposed  0.001246
3406  power consumption  0.001240
3570  propose novel  0.001239
4945  widely used  0.001237
448   based approach  0.001234
1517  energy consumption  0.001233
2969  new method  0.001232
2968  new approach  0.001224

```

```

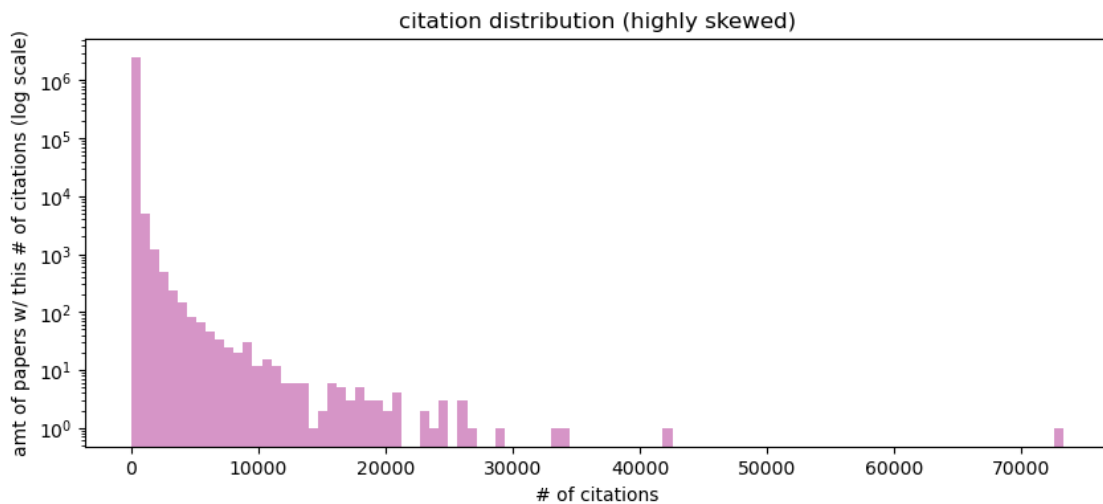
4471      support vector  0.001217
3584      proposed scheme 0.001209
2449      large number  0.001201

```

```

[11]: dblp["n_citation"].describe()
plt.figure(figsize=(10,4))
plt.hist(dblp["n_citation"], bins=100, log=True, color="#d695c7")
plt.xlabel("# of citations")
plt.ylabel("amt of papers w/ this # of citations (log scale)")
plt.title("citation distribution (highly skewed)")
plt.show()

```



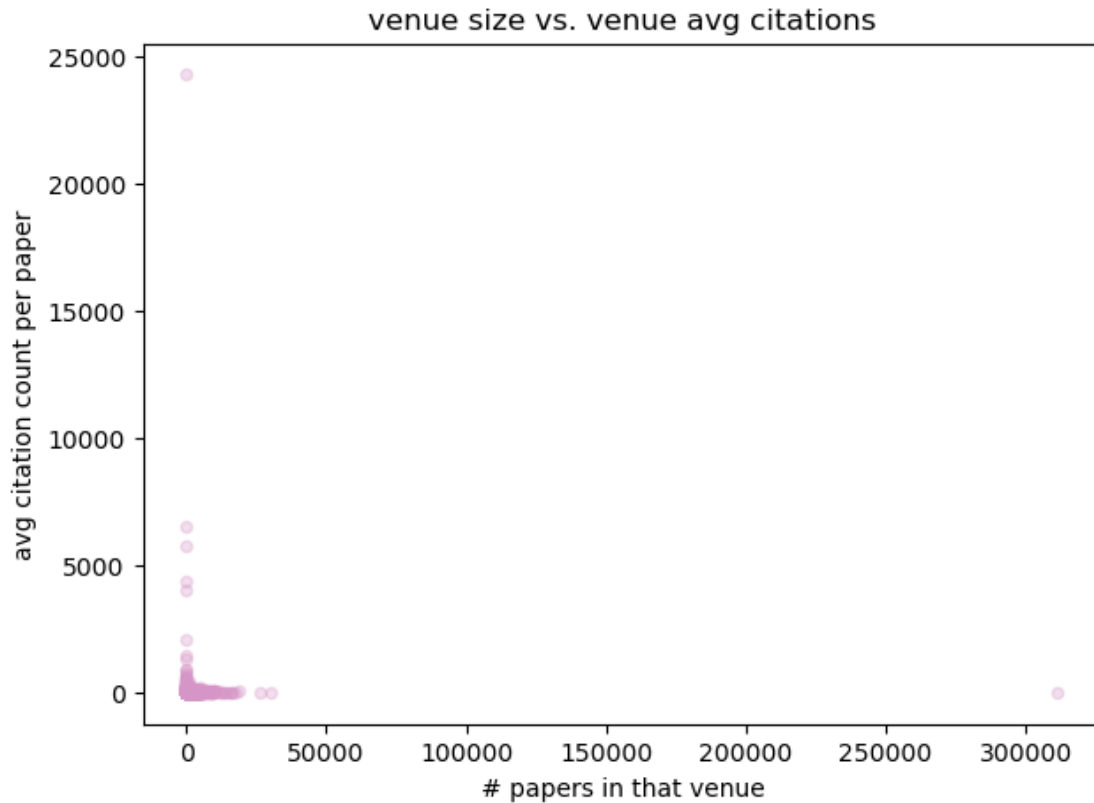
```

[12]: venuebycit = (
    dblp.groupby("venue")["n_citation"]
        .agg(
            **{
                "# papers in that venue": "count",
                "avg citation count per paper": "mean",
            }
        )
)

venuebycit.plot.scatter(
    x="# papers in that venue",
    y="avg citation count per paper",
    alpha=0.3,
    figsize=(7, 5),
    color="#d695c7"
)

```

```
plt.title("venue size vs. venue avg citations")
plt.xlabel("# papers in that venue")
plt.ylabel("avg citation count per paper")
plt.show()
```

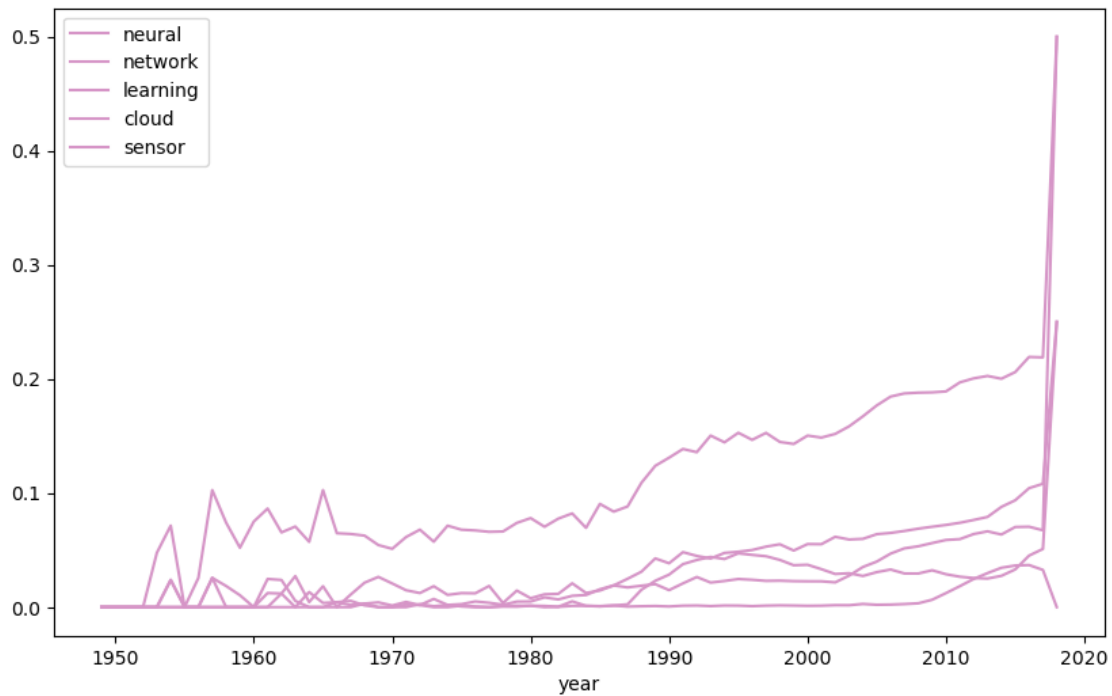


```
[13]: keywords = ["neural", "network", "learning", "cloud", "sensor"]

trend_df = pd.DataFrame({
    kw: dblp["text"].str.contains(kw).groupby(dblp["year"]).mean()
    for kw in keywords
})

trend_df.plot(figsize=(10,6), color="#d695c7")
```

```
[13]: <Axes: xlabel='year'>
```



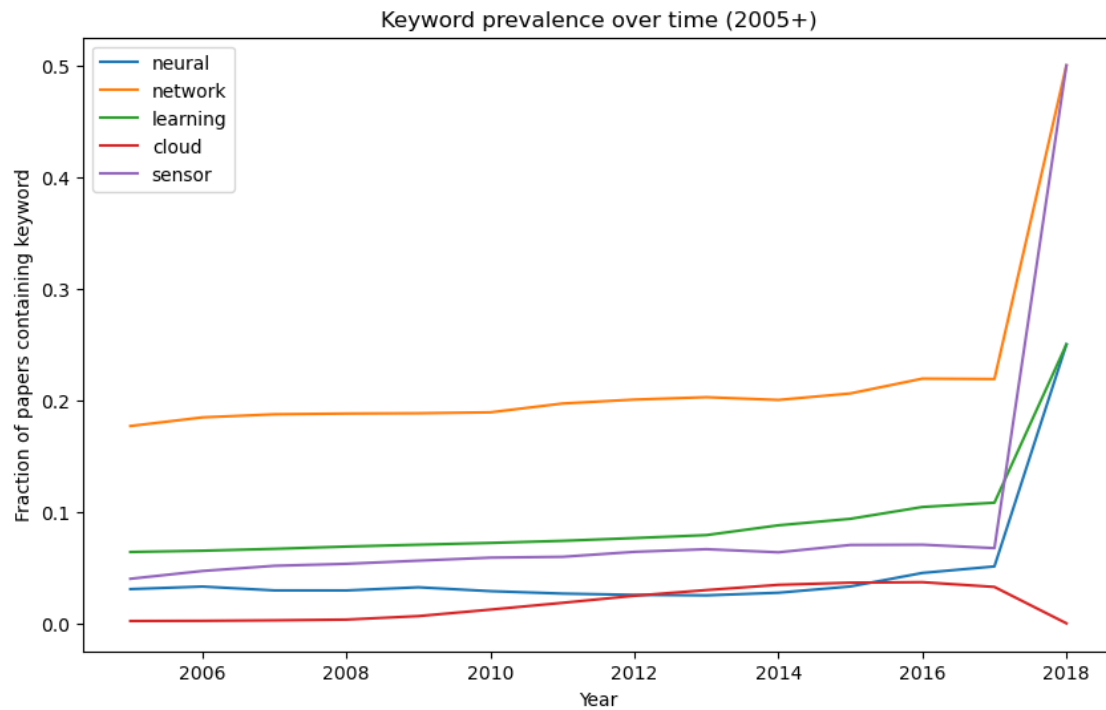
```
[ ]: keywords = ["neural", "network", "learning", "cloud", "sensor"]
```

```
dblp_2005 = dblp[dblp["year"] >= 2005].copy()
```

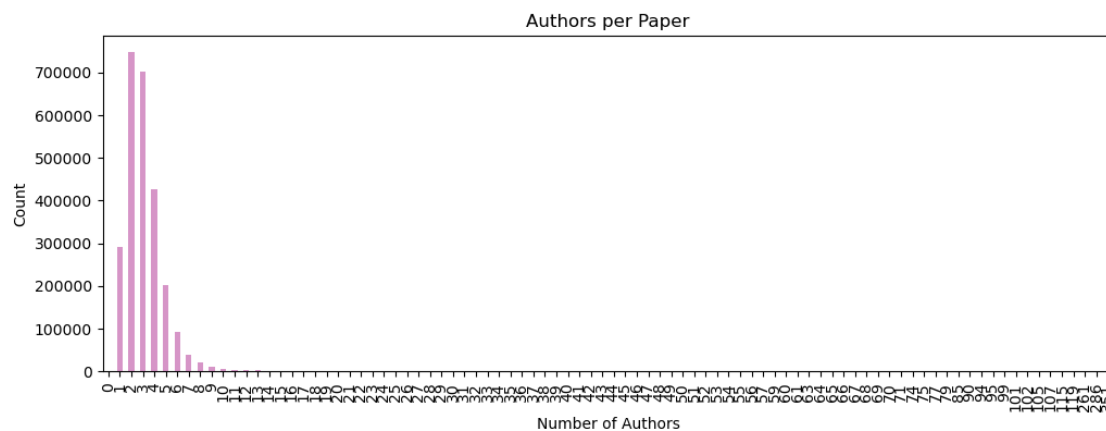
```
trend_df = pd.DataFrame({
    kw: dblp_2005["text"]
        .str.contains(kw)
        .groupby(dblp_2005["year"])
        .mean()
    for kw in keywords
})
```

```
trend_df.plot(figsize=(10, 6))
plt.title("Keyword prevalence over time (2005+)")
plt.xlabel("Year")
plt.ylabel("Fraction of papers containing keyword")
plt.show()
```

```
#same thing but after 2005 only
```



```
[15]: dblp["n_authors"].value_counts().sort_index().plot(kind="bar", figsize=(12,4),
        color="#d695c7")
plt.title("Authors per Paper")
plt.xlabel("Number of Authors")
plt.ylabel("Count")
plt.show()
```



```
[16]: from collections import Counter

author_counts = Counter(a for lst in dblp["authors"] for a in lst)
pd.DataFrame(author_counts.most_common(20), columns=["author", "papers"])

#probbaly due to them having the same name, not because the chinese are
↳dominating like that
#note that Lajos HAnzo was EIC of IEEE, which is why he's on there despite
↳being the only Lajos Hanzo
```

```
[16]:
```

	author	papers
0	Wei Wang	2261
1	Wei Zhang	1497
2	Lei Zhang	1438
3	Yang Liu	1346
4	Wei Li	1307
5	Jun Wang	1225
6	Lei Wang	1215
7	Jun Zhang	1147
8	Wei Liu	1057
9	Li Zhang	1019
10	Wei Chen	1013
11	Yan Zhang	1010
12	Xin Li	955
13	Yang Yang	930
14	Jing Wang	919
15	Wen Gao	915
16	Xin Wang	912
17	Lajos Hanzo	911
18	Yu Zhang	897
19	Jun Li	891

1.1 TASK #2: CITATION ANOMALY DETECTION

```
[ ]: import numpy as np
import pandas as pd

dblp_anom = dblp.dropna(subset=["venue"]).copy()

venue_stats_anom = (
    dblp_anom
    .groupby("venue")["n_citation"]
    .agg(["count", "mean", "std"])
    .reset_index()
)
```

```

min_papers_per_venue = 75
venue_stats_anom = venue_stats_anom[venue_stats_anom["count"] >=
    ↪ min_papers_per_venue]

#labeling as prestigious or unprestigious
q25 = venue_stats_anom["mean"].quantile(0.25)
q75 = venue_stats_anom["mean"].quantile(0.75)

def label_prestige(m):
    if m >= q75:
        return "prestigious"
    elif m <= q25:
        return "unprestigious"
    else:
        return "middle"

venue_stats_anom["prestige"] = venue_stats_anom["mean"].apply(label_prestige)

dblp_v = dblp_anom.merge(
    venue_stats_anom[["venue", "count", "mean", "std", "prestige"]],
    on="venue",
    how="inner"    # only venues that passed the count filter
)

#z-score of each paper's citations within its venue
dblp_v["std"] = dblp_v["std"].replace(0, np.nan) # avoid divide by 0
dblp_v["z_citation"] = (dblp_v["n_citation"] - dblp_v["mean"]) / dblp_v["std"]

#defining low and high
prestigious_low = (
    dblp_v[
        (dblp_v["prestige"] == "prestigious") &
        (dblp_v["z_citation"] <= -2.5)
    ]
    .sort_values("z_citation")
)

prestigious_low[
    ["venue", "year", "title", "n_citation", "mean", "z_citation"]
].head(20)

unprestigious_high = (
    dblp_v[
        (dblp_v["prestige"] == "unprestigious") &
        (dblp_v["z_citation"] >= 2.5)
    ]
)

```

```

    ]
    .sort_values("z_citation", ascending=False)
)

unprestigious_high[
    ["venue", "year", "title", "n_citation", "mean", "z_citation"]
].head(20)

prestigious_low["anomaly_type"] = "prestigious venue, low citation"
unprestigious_high["anomaly_type"] = "unprestigious venue, high citation"

all_anomalies = pd.concat([prestigious_low, unprestigious_high],
    ↪ignore_index=True)

all_anomalies[
    ["anomaly_type", "venue", "year", "title", "n_citation", "mean",
    ↪"z_citation"]
].head(50)

```

```

[ ]:
      anomaly_type \
0  unprestigious venue, high citation
1  unprestigious venue, high citation
2  unprestigious venue, high citation
3  unprestigious venue, high citation
4  unprestigious venue, high citation
5  unprestigious venue, high citation
6  unprestigious venue, high citation
7  unprestigious venue, high citation
8  unprestigious venue, high citation
9  unprestigious venue, high citation
10 unprestigious venue, high citation
11 unprestigious venue, high citation
12 unprestigious venue, high citation
13 unprestigious venue, high citation
14 unprestigious venue, high citation
15 unprestigious venue, high citation
16 unprestigious venue, high citation
17 unprestigious venue, high citation
18 unprestigious venue, high citation
19 unprestigious venue, high citation
20 unprestigious venue, high citation
21 unprestigious venue, high citation
22 unprestigious venue, high citation
23 unprestigious venue, high citation
24 unprestigious venue, high citation
25 unprestigious venue, high citation
26 unprestigious venue, high citation

```

27 unprestigious venue, high citation
 28 unprestigious venue, high citation
 29 unprestigious venue, high citation
 30 unprestigious venue, high citation
 31 unprestigious venue, high citation
 32 unprestigious venue, high citation
 33 unprestigious venue, high citation
 34 unprestigious venue, high citation
 35 unprestigious venue, high citation
 36 unprestigious venue, high citation
 37 unprestigious venue, high citation
 38 unprestigious venue, high citation
 39 unprestigious venue, high citation
 40 unprestigious venue, high citation
 41 unprestigious venue, high citation
 42 unprestigious venue, high citation
 43 unprestigious venue, high citation
 44 unprestigious venue, high citation
 45 unprestigious venue, high citation
 46 unprestigious venue, high citation
 47 unprestigious venue, high citation
 48 unprestigious venue, high citation
 49 unprestigious venue, high citation

	venue	year	\
0	arXiv: Networking and Internet Architecture	1998	
1	Proceedings of SPIE	1993	
2	Proceedings of SPIE	1995	
3	IEEE Access	2013	
4	arXiv: Information Theory	2007	
5	high performance computing and communications	2008	
6	Wireless Personal Communications	1997	
7	computational intelligence and security	2009	
8	computational science and engineering	1996	
9	arXiv: Distributed, Parallel, and Cluster Comp...	2016	
10	arXiv: Cryptography and Security	2009	
11	Procedia Computer Science	2013	
12	international geoscience and remote sensing sy...	1997	
13	Proceedings of SPIE	1995	
14	arXiv: Learning	2014	
15	arXiv: Combinatorics	1992	
16	arXiv: Computer Vision and Pattern Recognition	2012	
17	international conference on bioinformatics	1999	
18	arXiv: Logic in Computer Science	1998	
19	networked computing and advanced information m...	2009	
20	arXiv: Social and Information Networks	2011	
21	Wireless Personal Communications	2000	

22 international geoscience and remote sensing sy... 1994
 23 international conference on algorithms and arc... 2010
 24 human robot interaction 2007
 25 arXiv: Machine Learning 2015
 26 arXiv: Learning 2012
 27 european conference on information systems 2009
 28 International Journal of Manufacturing Technol... 2000
 29 advances in computing and communications 1994
 30 arXiv: Computer Vision and Pattern Recognition 2013
 31 IEICE Transactions on Information and Systems 2007
 32 IEICE Transactions on Electronics 2005
 33 arXiv: Computers and Society 1999
 34 emerging technologies and factory automation 2003
 35 advances in computing and communications 2010
 36 Ksii Transactions on Internet and Information ... 2016
 37 international symposium on signal processing a... 2007
 38 computer and information technology 2004
 39 Journal of Circuits, Systems, and Computers 1993
 40 arXiv: Discrete Mathematics 2008
 41 arXiv: Information Retrieval 2011
 42 Journal of Applied Mathematics 2011
 43 Wireless Personal Communications 2011
 44 arXiv: Learning 2013
 45 International Journal of Software Engineering ... 2001
 46 Journal of Machine Vision and Applications 1988
 47 international conference on human-computer int... 2003
 48 Kybernetika 1964
 49 arXiv: Computer Vision and Pattern Recognition 2016

	title	n_citation	mean \
0	A Quantitative Measure Of Fairness And Discrim...	3525	13.846186
1	The QBIC Project : Querying Images by Content ...	2175	14.884671
2	Similarity of color images	2116	14.884671
3	Millimeter Wave Mobile Communications for 5G C...	2231	10.963959
4	Physical Layer Network Coding	1066	13.508475
5	Market-Oriented Cloud Computing: Vision, Hype,...	2685	16.703358
6	Associativity-Based Routing for Ad Hoc Mobile ...	1298	18.720531
7	A detailed analysis of the KDD CUP 99 data set	1216	14.310506
8	Artificial neural networks: a tutorial	1806	18.471740
9	TensorFlow: Large-Scale Machine Learning on He...	2450	15.211284
10	Role-Based Access Controls	1148	13.705819
11	A Systems Approach Towards Reliability-Centred...	685	9.757464
12	A physics-based algorithm for retrieving land-...	748	13.491232
13	Techniques for data hiding	1527	14.884671
14	A Tutorial on Principal Component Analysis	1874	17.205582
15	Three-dimensional alpha shapes	1958	11.830594
16	UCF101: A Dataset of 101 Human Actions Classes...	825	12.428535

17	Identifying DNA and protein patterns with stat...	1318	15.521695
18	Stable Models and an Alternative Logic Program...	776	10.629784
19	A Taxonomy and Survey of Cloud Computing Systems	1316	14.469014
20	The Anatomy of the Facebook Social Graph	778	10.442228
21	Adaptive Modulation over Nakagami Fading Channels	842	18.720531
22	The effect of unlabeled samples in reducing th...	588	13.491232
23	InterCloud: utility-oriented federation of clo...	1018	16.323864
24	Human-robot interaction: a survey	894	7.983146
25	Distilling the Knowledge in a Neural Network	752	13.303344
26	ADADELTA: An Adaptive Learning Rate Method	1368	17.205582
27	RECONSTRUCTING THE GIANT: ON THE IMPORTANCE OF...	767	15.661290
28	Information sharing in a supply chain	1082	11.951945
29	Robust constrained model predictive control us...	455	13.855157
30	Deep Inside Convolutional Networks: Visualisin...	613	12.428535
31	A Speech Parameter Generation Algorithm Consid...	463	12.465057
32	Standby and Active Leakage Current Control and...	309	6.811849
33	Beyond Concern: Understanding Net Users' Attit...	521	10.878852
34	Survey on wireless sensor network devices	445	18.307642
35	Model predictive control for the operation of ...	421	13.855157
36	The MovieLens Datasets: History and Context	361	8.340309
37	A Leaf Recognition Algorithm for Plant Classif...	478	11.110535
38	Artificial intelligence with uncertainty	425	15.594801
39	CHAOS SYNCHRONIZATION IN CHUA'S CIRCUIT	401	8.612360
40	Combining geometry and combinatorics: A unifie...	360	8.738068
41	A new ANEW: Evaluation of a word list for sent...	434	12.162416
42	Mittag-Leffler Functions and Their Applications	339	8.008962
43	Internet of Things: Applications and Challenge...	594	18.720531
44	Playing Atari with Deep Reinforcement Learning	1107	17.205582
45	AGENT UML: A FORMALISM FOR SPECIFYING MULTIAGE...	869	18.446429
46	A New Scheme for Practical, Flexible and Intel...	355	10.670213
47	Location-Based Services for Mobile Telephony: ...	585	17.182171
48	Optomotorische Untersuchung des visuellen syst...	572	15.578947
49	Densely Connected Convolutional Networks	524	12.428535

z_citation

0	46.234605
1	40.628973
2	39.519260
3	39.088375
4	38.004527
5	37.080692
6	37.077224
7	36.080048
8	35.014356
9	32.793047
10	32.641567
11	32.029479

```

12  30.408261
13  28.440932
14  28.351715
15  27.084916
16  25.740413
17  25.213316
18  24.820294
19  24.582939
20  23.953094
21  23.861023
22  23.784349
23  22.912692
24  21.537671
25  20.860668
26  20.625514
27  19.813271
28  19.681650
29  19.451928
30  19.024736
31  18.834231
32  18.709887
33  18.129846
34  18.003012
35  17.952725
36  17.707594
37  17.072217
38  17.071257
39  16.846434
40  16.809907
41  16.775779
42  16.675595
43  16.673265
44  16.640260
45  16.470174
46  16.376564
47  16.345550
48  16.295526
49  16.205419

```

```

[18]: import numpy as np
import pandas as pd

dblp_cleaned = dblp.dropna(subset=["venue"]).copy()
dblp_cleaned["log_citations"] = np.log1p(dblp_cleaned["n_citation"])
#transforming to account for how massively skewed citations are so we can run Z_
↪score, find anomalies

```

```

venuedeets = (
    dblp_cleaned.groupby("venue")["log_citations"]
    .agg(venue_count=("count"), venue_log_mean=("mean"), venue_log_std=("std"))
    .reset_index()
)

min_papers = 75
venuedeets = venuedeets[venuedeets["venue_count"] >= min_papers].copy()

percentile50 = venuedeets["venue_log_mean"].quantile(0.50)
percentile85 = venuedeets["venue_log_mean"].quantile(0.85)

def get_prestige_label(val):
    if val <= percentile50:
        return "unprestigious"
    if val >= percentile85:
        return "prestigious"
    return "middle"

venuedeets["prestige_level"] = venuedeets["venue_log_mean"].
    ↪ apply(get_prestige_label)

paperswdeets = dblp_cleaned.merge(venuedeets, on="venue", how="inner")

paperswdeets["z_score"] = (
    (paperswdeets["log_citations"] - paperswdeets["venue_log_mean"])
    / paperswdeets["venue_log_std"].replace(0, np.nan)           #avoiding
    ↪ dividing by 0 if sd is 0 wrt finding Z score
)

#identifying anomalies now, epicly
is_prest_low = (paperswdeets["prestige_level"] == "prestigious") &
    ↪ (paperswdeets["z_score"] <= -2.5)
is_unprest_high = (paperswdeets["prestige_level"] == "unprestigious") &
    ↪ (paperswdeets["z_score"] >= 2.5)

anomalies = paperswdeets[is_prest_low | is_unprest_high].copy()
anomalies["anomaly_type"] = np.where(
    anomalies["z_score"] > 0,
    "unprestigious venue but high citations",
    "prestigious venue but low citations"
)

anomalies["|Z|"] = anomalies["z_score"].abs()
final_view = anomalies.sort_values("|Z|", ascending=False)

```

```
cols_to_show = ["anomaly_type", "venue", "year", "title", "n_citation", "z_score"]
final_view[cols_to_show].head(20)
```

```
[18]:
```

	anomaly_type \	venue	year \
343324	unprestigious venue but high citations		1989
1175386	prestigious venue but low citations		
1480583	unprestigious venue but high citations		1999
1781938	unprestigious venue but high citations	Datenschutz Und Datensicherheit	2002
349699	unprestigious venue but high citations		1996
1891286	unprestigious venue but high citations		1992
1988984	prestigious venue but low citations	computer security foundations workshop	1996
276361	unprestigious venue but high citations		1992
116824	prestigious venue but low citations	eurographics symposium on rendering techniques	2007
241057	prestigious venue but low citations	eurographics symposium on rendering techniques	1999
168114	prestigious venue but low citations	eurographics symposium on rendering techniques	2007
1568610	unprestigious venue but high citations	human robot interaction	2007
340889	unprestigious venue but high citations	arXiv: Networking and Internet Architecture	1998
348319	unprestigious venue but high citations		1974
147912	unprestigious venue but high citations		2003
729525	unprestigious venue but high citations	International Journal of Mathematical Modellin...	2010
195380	unprestigious venue but high citations		2008
646700	unprestigious venue but high citations	arXiv: Combinatorics	1992
1642530	unprestigious venue but high citations	IEEE Access	2013
143412	unprestigious venue but high citations		1989

	title	n_citation	\
343324	Genetic Algorithms in Search, Optimization and...	73362	
1175386	Semiregular patterns on surfaces	0	
1480583	Ad-hoc on-demand distance vector routing	26357	
1781938	Impact of artificial gummy fingers on fingerpr...	819	
349699	Handbook of Applied Cryptography	18201	
1891286	Genetic algorithms + data structures=evolution...	18006	
1988984	Action systems for security specification	0	
276361	Genetic programming: on the programming of com...	15096	
116824	Global illumination for the masses	0	
241057	Disruptive technologies in computer graphics: ...	0	
168114	The random camera, the coded aperture camera, ...	0	
1568610	Human-robot interaction: a survey	894	
340889	A Quantitative Measure Of Fairness And Discrim...	3525	
348319	The Design and Analysis of Computer Algorithms	13227	
147912	Iterative Methods for Sparse Linear Systems	13104	
729525	Engineering Optimisation by Cuckoo Search	1377	
195380	Introduction to Information Retrieval	12627	
646700	Three-dimensional alpha shapes	1958	
1642530	Millimeter Wave Mobile Communications for 5G C...	2231	
143412	Communication and concurrency	11519	

	z_score
343324	5.580289
1175386	-5.419102
1480583	4.970239
1781938	4.784723
349699	4.749592
1891286	4.743173
1988984	-4.669023
276361	4.638127
116824	-4.588400
241057	-4.588400
168114	-4.588400
1568610	4.567517
340889	4.565839
348319	4.559365
147912	4.553798
729525	4.551472
195380	4.531701
646700	4.513161
1642530	4.487414
143412	4.476973

analysis here I identified a data quality issue where X number of records had empty strings for venues. Upon inspection, these appeared to be Books or standalone works.

Also, the fact that some of these papers have 0 citations is very suspicious. I suspect it's an error on part of the dataset. However, the 'Unprestigious High' anomalies (like 'Genetic Algorithms') were verified as actual high-impact papers

```
[ ]: #here we run things for |z| > 2.5 to see if anomalous stuff has different TF
      ↳ IDF key terms than the whole population

if "text" not in dblp.columns:
    dblp["text"] = dblp["title"].fillna("") + " " + dblp["abstract"].fillna("")

#Merging. We use suffixes to prevent the "text_x" / "text_y" error if ran
↳ multiple times
anomalies_with_text = anomalies.merge(
    dblp[["text"]],
    left_index=True,
    right_index=True,
    how="inner",
    suffixes=("", "_new") # If text exists, the new one gets a suffix
)

#handle cases where the merge created a duplicate column
if "text_new" in anomalies_with_text.columns:
    anomalies_with_text["text"] = anomalies_with_text["text_new"]

#transform using the existing tfidf model
X_anom = tfidf.transform(anomalies_with_text["text"])
mean_tfidf_anom = X_anom.mean(axis=0).A1

#compare
comparison = pd.DataFrame({
    "term": terms,
    "general_score": mean_tfidf,
    "anomaly_score": mean_tfidf_anom
})

comparison["diff"] = comparison["anomaly_score"] - comparison["general_score"]
comparison = comparison[~comparison["term"].isin(boiler_phrases)]
bigrams_comp = comparison[comparison["term"].str.contains(" ")]

top_anomaly_terms = bigrams_comp.sort_values("diff", ascending=False).head(20)

print("\nTop Terms distinguishing Anomalies from the Rest:")
print(top_anomaly_terms[["term", "diff", "anomaly_score", "general_score"]].
      ↳ to_string(index=False))
```

```
Top Terms distinguishing Anomalies from the Rest:
      term      diff  anomaly_score  general_score
```

network coding	0.000425	0.000894	0.000469
object oriented	0.000419	0.001513	0.001094
simulation results	0.000418	0.003015	0.002596
video coding	0.000404	0.000907	0.000503
fourier transform	0.000403	0.000871	0.000469
linear systems	0.000359	0.000902	0.000543
time varying	0.000354	0.001444	0.001089
base station	0.000349	0.000908	0.000558
information systems	0.000321	0.001443	0.001122
based approach	0.000314	0.001547	0.001234
paper investigates	0.000290	0.000949	0.000660
content based	0.000285	0.000840	0.000555
resource management	0.000269	0.000703	0.000434
boundary conditions	0.000258	0.000629	0.000371
multi hop	0.000249	0.000724	0.000475
embedded systems	0.000245	0.000878	0.000634
query processing	0.000241	0.000601	0.000360
vector machines	0.000239	0.000754	0.000515
present novel	0.000230	0.001042	0.000812
closed loop	0.000226	0.000920	0.000694

```
[41]: import matplotlib.pyplot as plt
import seaborn as sns

#prepping w/ top 10 terms unique to anomalies (positive diff) + bottom 10 terms
↳typical of normal papers (negative diff)
top_15 = bigrams_comp.sort_values("diff", ascending=False).head(15)
bot_15 = bigrams_comp.sort_values("diff", ascending=True).head(15)
viz_data = pd.concat([top_15, bot_15])

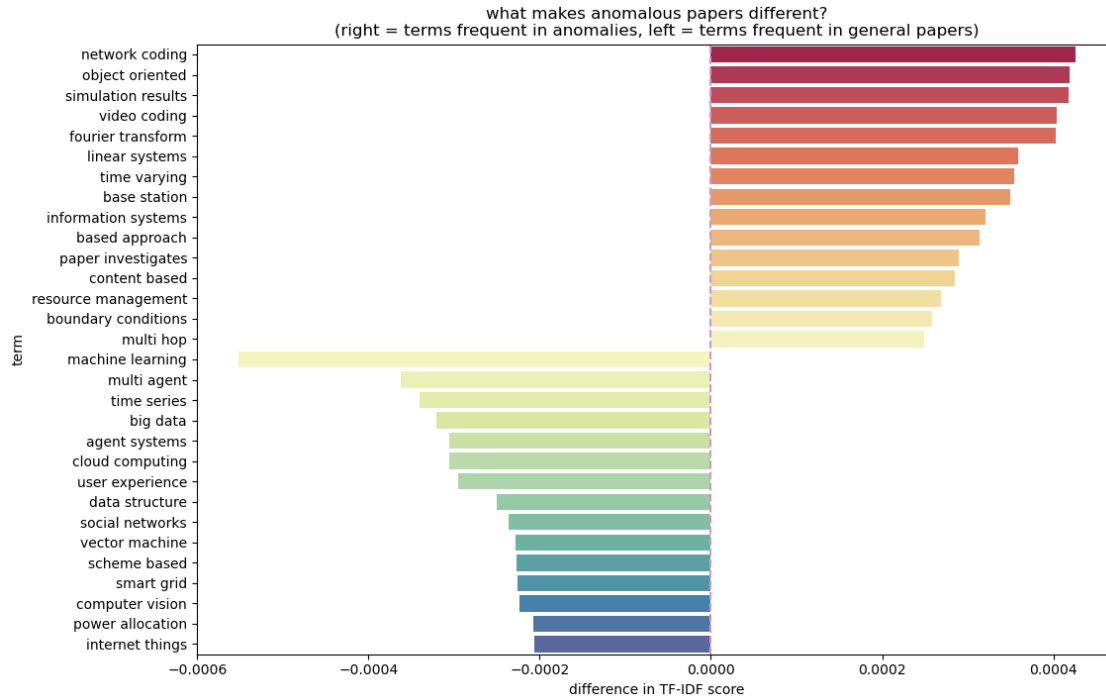
plt.figure(figsize=(12, 8))
sns.barplot(data=viz_data, x="diff", y="term", palette="Spectral")

plt.title("what makes anomalous papers different?\n(right = terms frequent in
↳anomalies, left = terms frequent in general papers)")
plt.xlabel("difference in TF-IDF score")
plt.axvline(0, color="#d695c7", linestyle="--")
plt.show()
```

C:\Users\Constance\AppData\Local\Temp\ipykernel_5464\3582927357.py:10:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

```
sns.barplot(data=viz_data, x="diff", y="term", palette="Spectral")
```



comments and analysis: machine learning being non-anomalous is both surprising and unsurprising. I suppose that since it's a hot topic, journals are picking it up, and it's likely to be "misplaced" into a journal of the wrong caliber compared to more obscure/rigorous fields like fourier transforms

2 TASK #3: Temporal Topic Analysis

```
[ ]: from sklearn.feature_extraction.text import TfidfVectorizer

dblp_temporal = dblp[dblp["year"] >= 1990]

samplesubset = (
    dblp_temporal
    .sample(200_000, random_state=2253221)
    .copy()
)

sample_text = samplesubset["text"].astype(str)
sample_year = samplesubset["year"].to_numpy()

tfidf_topics = TfidfVectorizer(
    max_features=5000,
    stop_words="english",
    min_df=50,
    max_df=0.4,
```

```

    ngram_range=(1, 2),
)

X_topics = tfidf_topics.fit_transform(sample_text)    #sparse mat
terms_topics = tfidf_topics.get_feature_names_out()

print("Shape:", X_topics.shape)
print("Example years:", sample_year[:10])
print("Example terms:", terms_topics[:10])

```

Shape: (200000, 5000)

Example years: [2006 1999 2004 2005 2005 2010 2004 2016 2010 1997]

Example terms: ['000' '10' '100' '11' '12' '13' '14' '15' '16' '17']

[22]: *#manually making terms for topics to put in the time series graph thingy*

```

topic_terms = {
    "Deep/Neural Learning": [
        "neural network", "neural networks",
        "deep learning", "convolutional neural",
        "recurrent neural",
    ],
    "Classical ML / SVM": [
        "support vector", "svm classifier",
        "kernel method", "kernel function",
    ],
    "Vision & Recognition": [
        "image segmentation", "object detection",
        "image classification", "face recognition",
    ],
    "Web & Semantic Web": [
        "semantic web", "web services",
        "linked data", "web mining",
        "information retrieval",
    ],
    "Sensor / Ad-hoc Networks": [
        "sensor networks", "wireless sensor",
        "ad hoc network", "routing protocol",
    ],
    "Cloud / Distributed": [
        "cloud computing", "grid computing",
        "map reduce", "distributed system",
    ],
    "Security & Crypto": [
        "access control", "authentication protocol",
        "public key", "cryptographic",
    ],
    "Data Mining & Recommenders": [

```

```

        "data mining", "frequent pattern",
        "association rules", "recommender system",
    ],
    "Optimization / RL": [
        "reinforcement learning", "dynamic programming",
        "model predictive", "control system",
    ],
    "NLP / Text": [
        "language model", "machine translation",
        "text classification",
    ],
    "Time Series & Forecasting": [
        "time series", "financial time", "stock market",
    ],
    "AI Alignment / Safety": [
        "alignment", "value alignment", "ai safety",
    ],
}

```

```

[ ]: import numpy as np
import pandas as pd

#term -> col index in X_topics
term_to_idx = {t: i for i, t in enumerate(terms_topics)}

topic_to_cols = {}
for topic, lex in topic_terms.items():
    cols = [term_to_idx[w] for w in lex if w in term_to_idx]
    if cols:
        topic_to_cols[topic] = cols

topic_to_cols #just to eyeball what survived

```

```

[ ]: {'Deep/Neural Learning': [2959, 2960],
      'Classical ML / SVM': [4478],
      'Vision & Recognition': [2155, 3024, 1722],
      'Web & Semantic Web': [4090, 4931, 2256],
      'Sensor / Ad-hoc Networks': [4105, 4962, 3974],
      'Cloud / Distributed': [729],
      'Security & Crypto': [83, 3625, 1083],
      'Data Mining & Recommenders': [1125, 370],
      'Optimization / RL': [3810, 1426],
      'Time Series & Forecasting': [4622],
      'AI Alignment / Safety': [223]}

```

```

[ ]: years = sample_year
years_unique = np.sort(np.unique(years))

```

```

rows_by_year = {y: np.where(years == y)[0] for y in years_unique}

topic_scores = []

for topic, cols in topic_to_cols.items():
    cols = np.array(cols)
    for y in years_unique:
        rows = rows_by_year[y]
        if len(rows) == 0:
            continue

        sub = X_topics[rows][:, cols]

        #per-capita: total TF-IDF for A topic in B yr / #docs B yr
        score = sub.sum() / len(rows)

        topic_scores.append({
            "topic": topic,
            "year": int(y),
            "score": float(score),
        })

topic_df = pd.DataFrame(topic_scores)
topic_df

```

```

[ ]:

```

	topic	year	score
0	Deep/Neural Learning	1990	0.003824
1	Deep/Neural Learning	1991	0.004716
2	Deep/Neural Learning	1992	0.006864
3	Deep/Neural Learning	1993	0.009908
4	Deep/Neural Learning	1994	0.005848
..
314	AI Alignment / Safety	2014	0.001470
315	AI Alignment / Safety	2015	0.001537
316	AI Alignment / Safety	2016	0.001212
317	AI Alignment / Safety	2017	0.000815
318	AI Alignment / Safety	2018	0.000000

[319 rows x 3 columns]

```

[25]: topic_pivot = (
        topic_df
        .pivot(index="year", columns="topic", values="score")
        .fillna(0)
    )

```

```
# focus on more recent years if you like
topic_pivot = topic_pivot[topic_pivot.index >= 2000]
topic_pivot = topic_pivot[topic_pivot.index != 2018]

topic_pivot.head(20)
```

```
[25]: topic  AI Alignment / Safety  Classical ML / SVM  Cloud / Distributed  \
year
2000          0.000870          0.000511          0.000000
2001          0.001180          0.000730          0.000000
2002          0.000881          0.000818          0.000000
2003          0.001564          0.000891          0.000000
2004          0.002056          0.001082          0.000000
2005          0.001566          0.001387          0.000000
2006          0.001536          0.001673          0.000000
2007          0.001411          0.001265          0.000000
2008          0.001352          0.001244          0.000075
2009          0.001396          0.001483          0.000526
2010          0.001615          0.001512          0.001241
2011          0.001714          0.001409          0.001391
2012          0.001745          0.001390          0.001994
2013          0.001531          0.001249          0.001618
2014          0.001470          0.001323          0.002120
2015          0.001537          0.001187          0.002261
2016          0.001212          0.001294          0.001562
2017          0.000815          0.001188          0.001351

topic  Data Mining & Recommenders  Deep/Neural Learning  Optimization / RL  \
year
2000          0.002516          0.004577          0.001179
2001          0.002078          0.006028          0.000881
2002          0.002322          0.004414          0.000803
2003          0.002007          0.004215          0.001136
2004          0.002537          0.003239          0.001081
2005          0.002419          0.005109          0.001144
2006          0.002137          0.005599          0.001025
2007          0.001957          0.004105          0.001383
2008          0.002219          0.003872          0.001146
2009          0.001881          0.004595          0.001272
2010          0.001668          0.004013          0.001155
2011          0.001809          0.003584          0.001237
2012          0.001670          0.002937          0.001207
2013          0.001417          0.002892          0.001006
2014          0.001222          0.003392          0.000976
2015          0.001301          0.003716          0.000964
2016          0.001038          0.005360          0.001076
```

2017	0.001117	0.005045	0.001373
topic	Security & Crypto	Sensor / Ad-hoc Networks	Time Series & Forecasting \
year			
2000	0.002713	0.000182	0.000950
2001	0.001869	0.000586	0.001703
2002	0.001882	0.000907	0.000709
2003	0.002515	0.001181	0.000920
2004	0.002583	0.002950	0.001008
2005	0.002679	0.004178	0.001072
2006	0.002448	0.004988	0.001456
2007	0.002455	0.005943	0.001209
2008	0.002444	0.005848	0.001015
2009	0.002579	0.006032	0.001439
2010	0.001955	0.005927	0.001340
2011	0.002469	0.005621	0.001295
2012	0.002015	0.005630	0.001610
2013	0.001914	0.005165	0.001458
2014	0.001716	0.004279	0.001572
2015	0.001979	0.004329	0.001626
2016	0.001821	0.003839	0.002085
2017	0.001805	0.004145	0.001450
topic	Vision & Recognition	Web & Semantic Web	
year			
2000	0.000855	0.001214	
2001	0.000751	0.001590	
2002	0.001915	0.002058	
2003	0.001595	0.004118	
2004	0.001620	0.004813	
2005	0.001608	0.005916	
2006	0.001554	0.004942	
2007	0.001561	0.004245	
2008	0.001833	0.004165	
2009	0.001921	0.003274	
2010	0.001618	0.003410	
2011	0.002106	0.002580	
2012	0.001779	0.002261	
2013	0.001681	0.001964	
2014	0.001730	0.002070	
2015	0.001573	0.001525	
2016	0.001951	0.001223	
2017	0.002060	0.000818	

```
[26]: import matplotlib.pyplot as plt

topics_to_plot = [
```

```

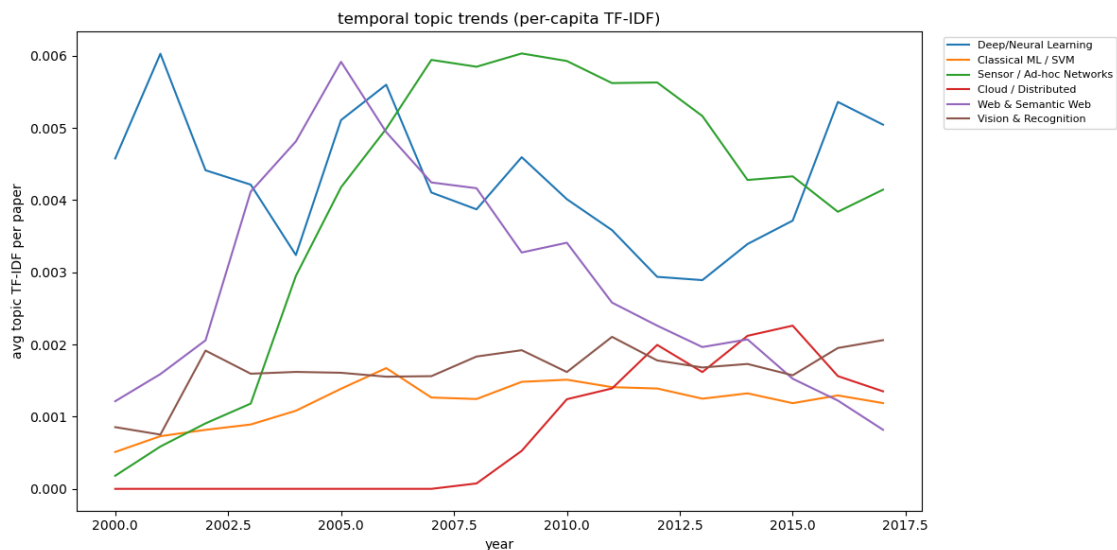
    "Deep/Neural Learning",
    "Classical ML / SVM",
    "Sensor / Ad-hoc Networks",
    "Cloud / Distributed",
    "Web & Semantic Web",
    "Vision & Recognition",
]

plt.figure(figsize=(12, 6))

for topic in topics_to_plot:
    if topic in topic_pivot.columns:
        plt.plot(topic_pivot.index, topic_pivot[topic], label=topic)

plt.xlabel("year")
plt.ylabel("avg topic TF-IDF per paper")
plt.title("temporal topic trends (per-capita TF-IDF)")
plt.legend(loc="upper left", bbox_to_anchor=(1.02, 1), fontsize=8)
plt.tight_layout()
plt.show()

```



```

[42]: from sklearn.model_selection import train_test_split
      from sklearn.linear_model import LogisticRegression
      from sklearn.metrics import classification_report, accuracy_score
      import numpy as np
      import pandas as pd

```

```

keep_indices = [i for i, term in enumerate(terms_topics) if not term.isdigit()]

X_clean = X_topics[:, keep_indices]
terms_clean = np.array(terms_topics)[keep_indices]

print(f"removed {X_topics.shape[1] - X_clean.shape[1]} numeric/year features.")
print(f"new feature count: {X_clean.shape[1]}")

y_era = (sample_year >= 2012).astype(int)

X_train, X_test, y_train, y_test = train_test_split(
    X_clean,
    y_era,
    test_size=0.2,
    random_state=2253221
)

clf = LogisticRegression(max_iter=1000)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
print("\naccuracy:", accuracy_score(y_test, y_pred))

print("\n--- log reg classificatroin report---")
print(classification_report(y_test, y_pred))

coefs = clf.coef_[0]
top_positive = np.argsort(coefs)[-10:]
top_negative = np.argsort(coefs)[:10]

print("\n--- top words (post-2012) ---")
print(terms_clean[top_positive])

print("\n--- top words (pre-2012) ---")
print(terms_clean[top_negative])

```

removed 60 numeric/year features.

new feature count: 4940

accuracy: 0.7095

--- log reg classificatroin report---

	precision	recall	f1-score	support
0	0.73	0.85	0.78	24596
1	0.67	0.49	0.57	15404

accuracy			0.71	40000
macro avg	0.70	0.67	0.67	40000
weighted avg	0.70	0.71	0.70	40000

--- top words (post-2012) ---

```
['deep' 'social media' 'state art' 'big' 'smartphone' 'iot' 'analytics'
 'cloud' 'sdn' 'big data']
```

--- top words (pre-2012) ---

```
['spl' 'cdma' 'sup' 'atm' 'mpeg' 'described' 'abstract' 'sensor networks'
 'computers' 'vlsi']
```

```
[44]: import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

#####c ase study identifier

indices_all = np.arange(len(sample_year))
_, indices_test, _, _ = train_test_split(
    indices_all,
    y_era,
    test_size=0.2,
    random_state=2253221
)

fp_mask = (y_test == 0) & (y_pred == 1)
fp_indices_local = np.where(fp_mask)[0]

print(f"Total False Positives available to inspect: {len(fp_indices_local)}")

if len(fp_indices_local) > 0:
    local_idx = np.random.choice(fp_indices_local)

    #map back to original dataframe index
    original_idx = indices_test[local_idx]

    print(f"\n--- randomly sel FP---")
    print(f"OG row index: {original_idx}")
    print(f"actual yr:      {sample_year[original_idx]} (Pre-2012)")

    #try block handles if sample_text is a series, except if numpy array
    try:
        text_content = sample_text.iloc[original_idx]
    except:
        text_content = sample_text[original_idx]
```

```

print(f"\n--- abstract ---\n{text_content}")

suspicious_words = ['cloud', 'deep', 'social', 'smart', 'big', 'analytics',
                    'network', 'data', 'mobile', 'web']

found_triggers = [w for w in suspicious_words if w in text_content.lower()]
print(f"\n--- potential triggers ---")
print(f"The model may have been confused by: {found_triggers}")

```

Total False Positives available to inspect: 3793

--- randomly sel FP---

OG row index: 96370

actual yr: 2003 (Pre-2012)

--- abstract ---

study on direct perception of collision avoidance mediated by brightness differences generally, ecological approaches toward obstacle avoidance employ "optic flow" as visual information. however, it is difficult to apply those approaches to navigations of a robot in the "real world", since optic flow is obtained through a complex image processing. in this paper, we proposed a method based on brightness differentials which are obtained by a simple process. using brightness differentials as an intrinsic metrics, robots perceive "affordance" from obstacles and can avoid them with the least effort.

--- potential triggers ---

The model may have been confused by: []

```

[ ]: from sklearn.ensemble import RandomForestClassifier

#n_jobs=-1 uses all processors to speed it up
rf_clf = RandomForestClassifier(n_estimators=100, random_state=2253221,
                               n_jobs=-1)

rf_clf.fit(X_train, y_train)

y_pred_rf = rf_clf.predict(X_test)

print("\n--- RF results ---")
print("accuracy:", accuracy_score(y_test, y_pred_rf))
print("\nclassification report:\n", classification_report(y_test, y_pred_rf))

acc_log = accuracy_score(y_test, y_pred) #assumes ypred from logreg
acc_rf = accuracy_score(y_test, y_pred_rf)

print(f"Logistic Regression accuracy: {acc_log:.4f}")

```

```
print(f"Random Forest accuracy:      {acc_rf:.4f}")
print(f"difference:                  {acc_rf - acc_log:.4f}")
```

Training Random Forest... (this might take 1-2 minutes)

--- RANDOM FOREST RESULTS ---

Accuracy: 0.691525

Classification Report:

	precision	recall	f1-score	support
0	0.69	0.91	0.78	24596
1	0.70	0.35	0.46	15404
accuracy			0.69	40000
macro avg	0.70	0.63	0.62	40000
weighted avg	0.69	0.69	0.66	40000

Logistic Regression Accuracy: 0.7095

Random Forest Accuracy: 0.6915

Difference: -0.0180

3 TASK #4: AUTHOR COMMUNITY CLUSTERING

Group authors into distinct research communities based on their co-authorship patterns and/or the content of their publications

```
[ ]: import numpy as np
import pandas as pd
from collections import Counter, defaultdict
import itertools
from scipy.sparse import lil_matrix

from sklearn.cluster import AgglomerativeClustering, SpectralClustering

dblp_auth = (
    dblp
    .dropna(subset=["authors", "year", "n_citation", "text"])
    .copy()
)

dblp_auth["authors"] = dblp_auth["authors"].apply(
    lambda x: x if isinstance(x, list) else []
)

#standardizing types
dblp_auth["year"] = dblp_auth["year"].astype(int)
```

```
dblp_auth["n_citation"] = dblp_auth["n_citation"].astype(int)

print("Rows in dblp_auth:", len(dblp_auth))
```

Rows in dblp_auth: 2548532

```
[ ]: # ----- per-author stats -----

exploded = (
    dblp_auth
    .explode("authors")
    .rename(columns={"authors": "author"})
)
exploded = exploded[exploded["author"].notna() & (exploded["author"] != "")]

author_stats = (
    exploded
    .groupby("author")
    .agg(
        n_papers=("author", "size"),
        total_citations=("n_citation", "sum"),
        first_year=("year", "min"),
        last_year=("year", "max"),
    )
)

author_stats["career_span"] = (
    author_stats["last_year"] - author_stats["first_year"] + 1
)

#distinct coauthors for each author
coauthor_sets = defaultdict(set)
for auth_list in dblp_auth["authors"]:
    for a in auth_list:
        if not a:
            continue
        coauthor_sets[a].update(x for x in auth_list if x and x != a)

author_stats["n_coauthors"] = author_stats.index.to_series().map(
    lambda a: len(coauthor_sets.get(a, set()))
)

author_counts = author_stats["n_papers"].to_dict()

author_stats
```

```
[ ]:
```

	n_papers	total_citations	first_year	\
author				
"lk" G"rler	1	50	2002	
(Alex) Chao-Chiang Meng	1	10	1991	
(TYPE=name) (SCHEME=Vancouver) Kahn Ce	20	610	1999	
-Jr. Paulo Drews	1	19	2013	
-Mali\$#353	1	50	2016	
...	
	1	0	2014	
	1	0	2004	
	1	0	2015	
	1	0	2011	
	1	0	2015	

	last_year	career_span	n_coauthors
author			
"lk" G"rler	2002	1	1
(Alex) Chao-Chiang Meng	1991	1	1
(TYPE=name) (SCHEME=Vancouver) Kahn Ce	2016	18	35
-Jr. Paulo Drews	2013	1	4
-Mali\$#353	2016	1	6
...
	2014	1	4
	2004	1	6
	2015	1	5
	2011	1	4
	2015	1	5

[1591444 rows x 6 columns]

```
[ ]: # ----- core authors -----

MIN_PAPERS = 30
MIN_COAUTHORS = 5
MIN_CITATIONS = 5000
MIN_SPAN = 7

elite_authors = author_stats[
    (author_stats["n_papers"]      >= MIN_PAPERS) &
    (author_stats["n_coauthors"]   >= MIN_COAUTHORS) &
    (author_stats["total_citations"] >= MIN_CITATIONS) &
    (author_stats["career_span"]   >= MIN_SPAN)
]

core_authors = set(elite_authors.index)

print("Number of elite/core authors:", len(core_authors))
```

```

#only w 2 core authors, so theres a web
def filter_core(auth_list):
    cores = [a for a in auth_list if a in core_authors]
    return cores if len(cores) >= 2 else None

dblp_auth["core_authors"] = dblp_auth["authors"].apply(filter_core)
dblp_core = dblp_auth.dropna(subset=["core_authors"]).copy()

print("# of papers with >=2 core authors:", len(dblp_core))

```

Number of elite/core authors: 6424
 # of papers with >=2 core authors: 175351

```

[ ]: # ----- co-occurrence matrix -----

core_authors_sorted = sorted(core_authors)
author_to_idx = {a: i for i, a in enumerate(core_authors_sorted)}
n_authors = len(core_authors_sorted)

A = lil_matrix((n_authors, n_authors), dtype=np.float32)

for auth_list in dblp_core["core_authors"]:
    idxs = [author_to_idx[a] for a in auth_list]
    for i, j in itertools.combinations(idxs, 2):
        A[i, j] += 1.0
        A[j, i] += 1.0

A.setdiag(0)
A = A.tocsr()

print("Co-occurrence matrix shape:", A.shape)

row_sums = np.array(A.sum(axis=1)).ravel()
nonzero_idx = np.where(row_sums > 0)[0]
print("Authors before:", A.shape[0], "after removing zero rows:",
      ↪len(nonzero_idx))

#deleting rows AND columns for zero-E authors
A_nz = A[nonzero_idx, :][:, nonzero_idx] # This is the key fix!
core_authors_nz = [core_authors_sorted[i] for i in nonzero_idx]
elite_stats_nz = elite_authors.loc[core_authors_nz]

print("A_nz shape:", A_nz.shape)

```

Co-occurrence matrix shape: (6424, 6424)
 Authors before: 6424 after removing zero rows: 6405
 A_nz shape: (6405, 6405)

```
[ ]: # ----- clustering: hierarchical + spectral -----

n_clusters = 8 #arbitrary

agg = AgglomerativeClustering(
    n_clusters=n_clusters,
    metric="cosine",
    linkage="average",
)

labels_hier = agg.fit_predict(A_nz.toarray())

spec = SpectralClustering(
    n_clusters=n_clusters,
    affinity="nearest_neighbors",
    n_neighbors=10,
    random_state=2253221,
    assign_labels="kmeans",
)

labels_spec = spec.fit_predict(A_nz)

print("len(core_authors_nz):", len(core_authors_nz))
print("len(labels_hier):", len(labels_hier))
print("len(labels_spec):", len(labels_spec))

author_clusters = pd.DataFrame({
    "author": core_authors_nz,
    "n_papers": elite_stats_nz["n_papers"].values,
    "total_citations": elite_stats_nz["total_citations"].values,
    "career_span": elite_stats_nz["career_span"].values,
    "n_coauthors": elite_stats_nz["n_coauthors"].values,
    "cluster_hier": labels_hier,
    "cluster_spec": labels_spec,
})

author_clusters.head()
```

c:\Users\Constance\anaconda3\envs\sklearn-env\Lib\site-packages\sklearn\cluster_agglomerative.py:584: ClusterWarning: The symmetric non-negative hollow observation matrix looks suspiciously like an uncondensed distance matrix

```
out = hierarchy.linkage(X, method=linkage, metric=affinity)
```

c:\Users\Constance\anaconda3\envs\sklearn-env\Lib\site-packages\sklearn\cluster_spectral.py:706: UserWarning: The spectral clustering API has changed. ``fit`` now constructs an affinity matrix from data. To use a custom affinity matrix, set ``affinity=precomputed``.

```
warnings.warn(
```

```
len(core_authors_nz): 6405
len(labels_hier): 6405
len(labels_spec): 6405
```

```
c:\Users\Constance\anaconda3\envs\sklearn-env\Lib\site-
packages\threadpoolctl.py:1226: RuntimeWarning:
Found Intel OpenMP ('libiomp') and LLVM OpenMP ('libomp') loaded at
the same time. Both libraries are known to be incompatible and this
can cause random crashes or deadlocks on Linux when loaded in the
same Python program.
Using threadpoolctl may cause crashes or deadlocks. For more
information and possible workarounds, please see
https://github.com/joblib/threadpoolctl/blob/master/multiple\_openmp.md
```

```
warnings.warn(msg, RuntimeWarning)
```

```
[ ]:
      author  n_papers  total_citations  career_span  n_coauthors  \
0  A. Ardeshir Goshtasby      54           6133          31           42
1      A. Del Bimbo      142           6981          30           76
2      A. E. Eiben      101           5152          24           98
3      A. K. Qin       47           6822          14           77
4      A. Murat Tekalp     110           6125          27          107

      cluster_hier  cluster_spec
0              0              0
1              0              0
2              0              0
3              0              0
4              0              0
```

```
[ ]: from sklearn.metrics import silhouette_score
import numpy as np

A_dense = A_nz.toarray() if hasattr(A_nz, 'toarray') else A_nz
max_val = A_dense.max()
A_dist = max_val - A_dense

np.fill_diagonal(A_dist, 0)

silhouette_scores = []
k_values = [2, 3, 5, 6, 8, 10, 12, 15, 18, 20, 25, 30, 40, 50, 75, 100]

for k in k_values:
    spec = SpectralClustering(
        n_clusters=k,
        affinity="precomputed",
        random_state=42,
        assign_labels="kmeans",
```

```

    )
    labels = spec.fit_predict(A_nz)
    sil_score = silhouette_score(A_dist, labels, metric='precomputed')
    silhouette_scores.append(sil_score)
    print(f"k={k}: silhouette={sil_score:.4f}")

# Plot the results
import matplotlib.pyplot as plt
plt.plot(k_values, silhouette_scores, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Score vs Number of Clusters')
plt.show()

# Choose the best k
best_k = k_values[silhouette_scores.index(max(silhouette_scores))]
print(f"Best k by silhouette score: {best_k}")

```

```

c:\Users\Constance\anaconda3\envs\sklearn-env\Lib\site-
packages\sklearn\manifold\_spectral_embedding.py:328: UserWarning: Graph is not
fully connected, spectral embedding may not work as expected.

```

```
warnings.warn(
```

```
k=2: silhouette=0.0000
```

```

c:\Users\Constance\anaconda3\envs\sklearn-env\Lib\site-
packages\sklearn\manifold\_spectral_embedding.py:328: UserWarning: Graph is not
fully connected, spectral embedding may not work as expected.

```

```
warnings.warn(
```

```
k=3: silhouette=0.0001
```

```

c:\Users\Constance\anaconda3\envs\sklearn-env\Lib\site-
packages\sklearn\manifold\_spectral_embedding.py:328: UserWarning: Graph is not
fully connected, spectral embedding may not work as expected.

```

```
warnings.warn(
```

```
k=5: silhouette=0.0003
```

```

c:\Users\Constance\anaconda3\envs\sklearn-env\Lib\site-
packages\sklearn\manifold\_spectral_embedding.py:328: UserWarning: Graph is not
fully connected, spectral embedding may not work as expected.

```

```
warnings.warn(
```

```
k=6: silhouette=0.0004
```

```

c:\Users\Constance\anaconda3\envs\sklearn-env\Lib\site-
packages\sklearn\manifold\_spectral_embedding.py:328: UserWarning: Graph is not
fully connected, spectral embedding may not work as expected.

```

```
warnings.warn(
```

```
k=8: silhouette=0.0005
```

```

c:\Users\Constance\anaconda3\envs\sklearn-env\Lib\site-
packages\sklearn\manifold\_spectral_embedding.py:328: UserWarning: Graph is not
fully connected, spectral embedding may not work as expected.
    warnings.warn(

k=10: silhouette=0.0005

c:\Users\Constance\anaconda3\envs\sklearn-env\Lib\site-
packages\sklearn\manifold\_spectral_embedding.py:328: UserWarning: Graph is not
fully connected, spectral embedding may not work as expected.
    warnings.warn(

k=12: silhouette=0.0006

c:\Users\Constance\anaconda3\envs\sklearn-env\Lib\site-
packages\sklearn\manifold\_spectral_embedding.py:328: UserWarning: Graph is not
fully connected, spectral embedding may not work as expected.
    warnings.warn(

k=15: silhouette=0.0008

c:\Users\Constance\anaconda3\envs\sklearn-env\Lib\site-
packages\sklearn\manifold\_spectral_embedding.py:328: UserWarning: Graph is not
fully connected, spectral embedding may not work as expected.
    warnings.warn(

k=18: silhouette=0.0009

c:\Users\Constance\anaconda3\envs\sklearn-env\Lib\site-
packages\sklearn\manifold\_spectral_embedding.py:328: UserWarning: Graph is not
fully connected, spectral embedding may not work as expected.
    warnings.warn(

k=20: silhouette=0.0009

c:\Users\Constance\anaconda3\envs\sklearn-env\Lib\site-
packages\sklearn\manifold\_spectral_embedding.py:328: UserWarning: Graph is not
fully connected, spectral embedding may not work as expected.
    warnings.warn(

k=25: silhouette=0.0012

c:\Users\Constance\anaconda3\envs\sklearn-env\Lib\site-
packages\sklearn\manifold\_spectral_embedding.py:328: UserWarning: Graph is not
fully connected, spectral embedding may not work as expected.
    warnings.warn(

k=30: silhouette=0.0013

c:\Users\Constance\anaconda3\envs\sklearn-env\Lib\site-
packages\sklearn\manifold\_spectral_embedding.py:328: UserWarning: Graph is not
fully connected, spectral embedding may not work as expected.
    warnings.warn(

k=40: silhouette=0.0016

```

```
c:\Users\Constance\anaconda3\envs\sklearn-env\Lib\site-  
packages\sklearn\manifold\_spectral_embedding.py:328: UserWarning: Graph is not  
fully connected, spectral embedding may not work as expected.
```

```
warnings.warn(  

```

```
k=50: silhouette=0.0018
```

```
c:\Users\Constance\anaconda3\envs\sklearn-env\Lib\site-  
packages\sklearn\manifold\_spectral_embedding.py:328: UserWarning: Graph is not  
fully connected, spectral embedding may not work as expected.
```

```
warnings.warn(  

```

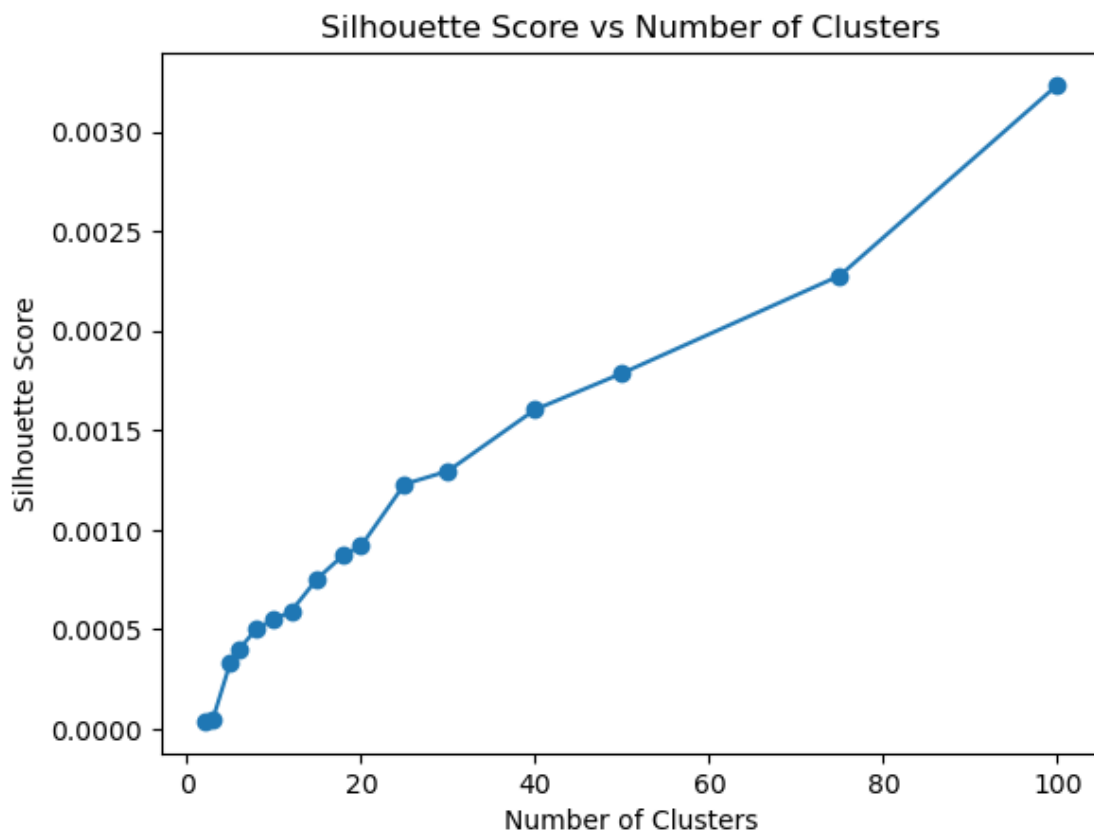
```
k=75: silhouette=0.0023
```

```
c:\Users\Constance\anaconda3\envs\sklearn-env\Lib\site-  
packages\sklearn\manifold\_spectral_embedding.py:328: UserWarning: Graph is not  
fully connected, spectral embedding may not work as expected.
```

```
warnings.warn(  

```

```
k=100: silhouette=0.0032
```



Best k by silhouette score: 100

```
[ ]: topic_terms = {
    "Deep/Neural Learning": [
        "deep learning", "neural network", "convolutional neural", "recurrent_
        ↪neural",
    ],
    "Classical ML / SVM": [
        "support vector machine", "svm", "kernel method", "naive bayes",
    ],
    "Vision & Recognition": [
        "image segmentation", "object detection", "computer vision",
    ],
    "Web & Semantic Web": [
        "semantic web", "ontology", "linked data", "web service",
    ],
    "Sensor / Ad-hoc Networks": [
        "sensor network", "wireless sensor", "ad hoc network",
    ],
    "Cloud / Distributed": [
        "cloud computing", "distributed system", "mapreduce",
    ],
    "Security & Crypto": [
        "intrusion detection", "cryptographic", "access control",
    ],
    "Data Mining & Recommenders": [
        "data mining", "recommendation system", "recommender system",
    ],
    "Optimization / RL": [
        "reinforcement learning", "stochastic optimization",
    ],
    "Time Series & Forecasting": [
        "time series", "forecasting", "temporal sequence",
    ],
    "AI Alignment / Safety": [
        "ai safety", "robustness", "fairness", "bias mitigation",
    ],
    "NLP / Text": [
        "natural language processing", "language model", "text mining",
    ],
}
```

```
[ ]: # ----- map authors to their papers -----

author_to_papers = {a: set() for a in core_authors_nz}

for idx, auth_list in dblp_core["core_authors"].items():
    for a in auth_list:
        if a in author_to_papers:
```

```

        author_to_papers[a].add(idx)

# convenient access to text for those papers
paper_text = dblp_core["text"].astype(str)
paper_text_lower = paper_text.str.lower()

```

```

[ ]: import re

def topic_profile_for_cluster(cluster_col, cluster_id):
    cluster_authors = author_clusters.loc[
        author_clusters[cluster_col] == cluster_id, "author"
    ].tolist()

    #all papers written by these authors
    paper_ids = set()
    for a in cluster_authors:
        paper_ids |= author_to_papers.get(a, set())

    if not paper_ids:
        return None, 0, cluster_authors

    texts = paper_text_lower.loc[list(paper_ids)].tolist()
    n_docs = len(texts)

    topic_scores = {}
    for topic, phrases in topic_terms.items():
        phrases_lower = [p.lower() for p in phrases]
        hits = 0
        for t in texts:
            if any(p in t for p in phrases_lower):
                hits += 1
        topic_scores[topic] = hits / n_docs #fraction of docs mentioning topic

    sorted_topics = sorted(topic_scores.items(), key=lambda x: x[1],
↪reverse=True)
    return sorted_topics, n_docs, cluster_authors

```

```

[39]: # ----- cluster summaries: hierarchical -----

cluster_summaries_hier = []

for c in sorted(author_clusters["cluster_hier"].unique()):
    topic_list, n_docs, cluster_authors =
↪topic_profile_for_cluster("cluster_hier", c)
    if topic_list is None:
        continue
    top3 = topic_list[:3]

```

```

cluster_summaries_hier.append({
    "cluster_hier": c,
    "n_authors": len(cluster_authors),
    "n_papers": n_docs,
    "top_topics": [t for t, score in top3],
    "top_topic_scores": [float(score) for t, score in top3],
})

cluster_summaries_hier_df = pd.DataFrame(cluster_summaries_hier)
cluster_summaries_hier_df

```

```

[39]:   cluster_hier  n_authors  n_papers  \

0          0         6397    175347
1          1           2         7
2          2           1        26
3          3           1         4
4          4           1         1
5          5           1         4
6          6           1         8
7          7           1        51

                                     top_topics  \
0  [Sensor / Ad-hoc Networks, AI Alignment / Safe...
1  [Deep/Neural Learning, Classical ML / SVM, Vis...
2  [Deep/Neural Learning, Classical ML / SVM, Opt...
3  [Deep/Neural Learning, Classical ML / SVM, Vis...
4  [Deep/Neural Learning, Classical ML / SVM, Vis...
5  [Deep/Neural Learning, Classical ML / SVM, Vis...
6  [Deep/Neural Learning, Classical ML / SVM, Vis...
7  [Web & Semantic Web, Deep/Neural Learning, Cla...

                                     top_topic_scores
0  [0.042030944356048296, 0.032917586271792504, 0...
1  [0.0, 0.0, 0.0]
2  [0.11538461538461539, 0.038461538461538464, 0...
3  [0.0, 0.0, 0.0]
4  [0.0, 0.0, 0.0]
5  [0.0, 0.0, 0.0]
6  [0.25, 0.0, 0.0]
7  [0.0784313725490196, 0.0, 0.0]

```

```

[ ]: # ----- cluster summaries: spectral -----

cluster_summaries_spec = []

for c in sorted(author_clusters["cluster_spec"].unique()):

```

```

    topic_list, n_docs, cluster_authors = \
topic_profile_for_cluster("cluster_spec", c)
    if topic_list is None:
        continue
    top3 = topic_list[:3]
    cluster_summaries_spec.append({
        "cluster_spec": c,
        "n_authors": len(cluster_authors),
        "n_papers": n_docs,
        "top_topics": [t for t, score in top3],
        "top_topic_scores": [float(score) for t, score in top3],
    })

cluster_summaries_spec_df = pd.DataFrame(cluster_summaries_spec)
cluster_summaries_spec_df

```

```

[ ]:   cluster_spec  n_authors  n_papers  \
0           0         6282    173724
1           1           11         452
2           2           16         290
3           3           13        1310
4           4           10         275
5           5           19         604
6           6           25        3633
7           7           29        1544

                                     top_topics  \
0  [Sensor / Ad-hoc Networks, AI Alignment / Safe...
1  [Vision & Recognition, AI Alignment / Safety, ...
2  [Web & Semantic Web, NLP / Text, Data Mining &...
3  [Sensor / Ad-hoc Networks, Security & Crypto, ...
4  [AI Alignment / Safety, Deep/Neural Learning, ...
5  [Classical ML / SVM, Deep/Neural Learning, Opt...
6  [Data Mining & Recommenders, Sensor / Ad-hoc N...
7  [Cloud / Distributed, Sensor / Ad-hoc Networks...

                                     top_topic_scores
0  [0.042107020331099906, 0.03289125279178467, 0...
1  [0.04424778761061947, 0.0420353982300885, 0.02...
2  [0.1482758620689655, 0.05517241379310345, 0.02...
3  [0.1480916030534351, 0.08396946564885496, 0.04...
4  [0.02909090909090909, 0.007272727272727273, 0.0]
5  [0.1837748344370861, 0.06291390728476821, 0.05...
6  [0.08835672997522709, 0.037985136251032205, 0...
7  [0.048575129533678756, 0.04145077720207254, 0...

```

analysis: it seems that most authors are part of one large, inter-connected research community, and that a handful are in small, more isolated communities.