Massive Data Analysis

Instructor: Dr.Gholampour

Fall 2024

HW 1

Arman Yazdani - 400102255

Contents

1 Research & Theory **2**

1.1 Join example . . . 2

1.1.1 Map . . . 2

1.1.2 Reduce . . . 2

1.1.3 Cost . . . 3

1.2 RDD vs DataFrame . . . 4

1.3 Boosting PySpark . . . 4

2 Practical **6**

2.1 Reading the data . . . 6

2.1.1 parse the json string . . . 6

2.1.2 Create a function that extracts and lists all fields (e.g., title, abstract, etc.) from the parsed RDD. . . . 6

2.2 Preprocessing . . . 6

2.2.1 Identify and remove or impute any null values, especially in critical fields 6

2.2.2 Find and remove stopwords . . . 7

2.2.3 Find and remove useless characters . . . 7

2.2.4 Even more purity! . . . 8

2.3 Dataset Analysis . . . 8

2.3.1 How many articles exist in each category (e.g., hep-ph, math.co)? . . . 8

2.3.2 Which category has the most articles? . . . 9

2.3.3 What is the distribution of the number of authors per article? (e.g., what percentage of articles have 1 author, more than 3 authors?) . . . 9

2.3.4 Filter out the articles that have more than 3 authors and generate a list of their titles and authors. Display the first 10 results. . . . 9

2.3.5 Plot a time series of the number of articles submitted per year. . . . 10

2.3.6 What are the 20 most frequent words in abstract? + 2.3.7: WordCloud 10

2.4 Advanced Data Exploration . . . 11

1 Research & Theory

1.1 Join example

Assume we want to analyze the join problem as follows:

R(A,B) S(B,C) T(C,D) U(D,E)

In this problem R,S,T,U are tables of size r,s,t,u, respectively. The probability that R&S agree on B, S&T agree on C, and T&U agree on D are all p.

• Using the Map-Reduce model and the **Single Step approach**, design an algorithm for solving this join problem.

• Evaluate the solution of this algorithm in terms of **Replication rate**, **Communication cost**, **Reducer size**, and the **number of Map and Reduce nodes**.

1.1.1 Map

We use the appropriate hash function. The hash function we use is the remainder of the division. Suppose *the number of reducers is* k. Keys are (i, j, k).

Let x, y, and z be three numbers such that $M = xyz$.

Map operation on the data:

```
For each row in R:
 For j in range(y):
  For k in range(z):
   emit <key=(mod(b, x), j, k), value = (a, b, 'R')>
For each row in S:
 For k in range(z):
  emit <key=(mod(b, x), mod(c, y), k), value=(b, c, 'S')>
For each row in T:
 For i range(x):
  emit <key=(i, mod(c, y), mod(d, z), value=(c, d, 'T'))>
For each row in U:
 For i in range(x):
  For j in range(y):
   emit <key = (i, j, mod(d, z), value(d, e, 'U'))>
```

1.1.2 Reduce

- All elements that have the same key are directed to a reducer.
- Consider the value of the elements in each reducer.

- Starting with the elements of table R and b: match them with b elements of table S, and likewise for c and d of tables T and U.
- Multiply all the a and e that are matched by Cartesian.

$$\textit{Calculate Cartesian product of}: (a_1, a_2, \ldots, a_n) \times (e_1, e_2, \ldots, e_m)$$

1.1.3 Cost

$$\textit{Cost} = r + s + t + u + ryz + sz + tx + uxy$$
$$\textit{Such that: } xyz = M \textit{ ; M: number of reducer}$$

We should find z,x,y such that the cost is lowest:

$$\textit{Min } r + s + t + u + ryz + sz + tx + uxy$$
$$\textit{Subject to } xyz = M$$

Which is equivalent to the optimization problem:

$$\text{Minimize } ryz + sz + tx + uxy$$
$$\text{Subject to: } xyz = M$$

Using *Lagrange*, we have:

$$\mathcal{L}(x, y, z, \lambda) = ryz + sz + tx + uxy + \lambda(xyz - M)$$
$$\frac{\partial \mathcal{L}}{\partial x} = t + uy + \lambda yz = 0$$
$$\frac{\partial \mathcal{L}}{\partial y} = rz + ux + \lambda xz = 0$$
$$\frac{\partial \mathcal{L}}{\partial z} = ry + s + \lambda xy = 0$$
$$\frac{\partial \mathcal{L}}{\partial \lambda} = xyz - M = 0$$

For simplicity, let:

$$u = t = s = r$$

Which leads to:

$$\frac{\partial \mathcal{L}}{\partial x} = r + ry + \lambda yz = 0$$
$$\frac{\partial \mathcal{L}}{\partial y} = rz + rx + \lambda xz = 0$$
$$\frac{\partial \mathcal{L}}{\partial z} = ry + r + \lambda xy = 0$$
$$\frac{\partial \mathcal{L}}{\partial \lambda} = xyz - M = 0$$

Resulting:

$$x = \sqrt{M}, \quad z = \sqrt{M}, \quad y = 1$$

Final cost is:

$$\textbf{Single Step Cost} = 4r + 4r\sqrt{M}$$

1.2 RDD vs DataFrame

Feature	RDD	DataFrame
Abstraction Level	Low-level abstraction	High-level abstraction
Type Safety	Supports compile-time type safety	Limited type safety; more schema-based
Data Representation	Distributed collection of objects	Distributed collection of data organized into named columns, similar to a table in a relational database
Optimization	No built-in optimization	Optimized through Catalyst query optimizer and Tungsten execution engine
API	Provides transformations and actions using a more functional programming style (e.g., map, filter, reduce)	Provides domain-specific language (DSL) and SQL-like query capabilities
Serialization	Requires manual serialization and deserialization	Built-in optimized serializers for faster processing
Schema	No inherent schema support; can hold any type of data	Enforces schema, making it easier to manipulate structured data
Performance	Typically less efficient for complex queries	Generally more efficient due to optimizations
Ease of Use	Requires more complex code to achieve certain tasks	Simplifies code with higher-level abstractions and SQL-like syntax
Use Cases	Suitable for unstructured data and complex transformations	Ideal for structured data and SQL-like operations

1.3 Boosting PySpark

1. **Use DataFrame/Dataset over RDD**:
   DataFrames and Datasets provide a higher-level abstraction and benefit from optimizations like Project Tungsten and the Catalyst optimizer. They are generally more efficient than RDDs.

2. **Optimize Data Serialization**:
   Use efficient data formats like Parquet or ORC for data storage and processing. These formats are optimized for both storage and speed.

3. **Cache Data**:
   Use caching to store frequently accessed data in memory. This can significantly reduce the time taken for repeated data access. You can cache DataFrames using the ‘cache()‘ method.

4. **Minimize Shuffling**:
   Shuffling is an expensive operation that involves redistributing data across the cluster. Minimize shuffling by using operations like ‘coalesce()‘ instead of ‘repartition()‘ when reducing the number of partitions.

5. **Tune Spark Configurations**:
   Adjust Spark configurations such as the number of executors, executor memory, and core usage to match your cluster’s resources. This ensures that Spark utilizes the available resources efficiently.

6. **Avoid UDFs (User Defined Functions)**:
   UDFs can be slow because they are executed in Python and not optimized by Spark. Try to use built-in functions and DataFrame operations whenever possible.

7. **Reduce Logging Levels**:
   Disable unnecessary logging levels like DEBUG and INFO to reduce overhead.

8. **Use Efficient Data Structures**:
   Use appropriate data structures and algorithms that are optimized for your specific use case.

9. **Partition Data Properly**:
   Ensure that your data is partitioned correctly to balance the load across the cluster. This can help in parallelizing the processing and improving performance

10. **Profile and Monitor**:
    Use Spark’s built-in tools like the web UI and event logs to monitor and profile your application. This can help identify performance bottlenecks and areas for improvement.

2 Practical

2.1 Reading the data

2.1.1 parse the json string

After mounting drive files,using *.textFile* and *.map(lambda line: json.loads(line))* functions:


```
['{"id":"0704.0001","submitter":"Pavel Nadolsky","authors":"C. Bal\\\\\\'azs, E. L. Berger, P. M. Nadolsky, C.-P. Yuan","title":"Calculation of prompt
diphoton production cross sections at Tevatron and\\n  LHC energies","comments":"37 pages, 15 figures; published version","journal-
ref":"Phys.Rev.D76:013009,2007","doi":"10.1103/PhysRevD.76.013009","report-no":"ANL-HEP-PR-07-12","categories":"hep-ph","license":null,"abstract":"  A fully
differential calculation in perturbative quantum chromodynamics is\\npresented for the production of massive photon pairs at hadron colliders. All\\nnext-to-
leading order perturbative contributions from quark-antiquark,\\ngluon-(anti)quark, and gluon-gluon subprocesses are included, as well as\\nall-orders
resummation of initial-state gluon radiation valid at\\nnext-to-next-to-leading logarithmic accuracy. The region of phase space is\\nspecified in which the
calculation is most reliable. Good agreement is\\ndemonstrated with data from the Fermilab Tevatron, and predictions are made for\\nmore detailed tests with
CDF and DO data. Predictions are shown for\\ndistributions of diphoton pairs produced at the energy of the Large Hadron\\nCollider (LHC). Distributions of
the diphoton pairs from the decay of a Higgs\\nboson are contrasted with those produced from QCD processes at the LHC, showing\\nthat enhanced sensitivity to
the signal can be obtained with judicious\\nselection of events.\\n","versions":[{"version":"v1","created":"Mon, 2 Apr 2007 19:18:42 GMT"},
{"version":"v2","created":"Tue, 24 Jul 2007 20:10:27 GMT"}],"update_date":"2008-11-26","authors_parsed":[["Bal\\u00e1zs","C.",""],["Berger","E. L.",""],
["Nadolsky","P. M.",""],["Yuan","C. -P.",""]]}']
```


Figure 1: Raw json(sample)


```
[{'id': '0704.0001',
  'submitter': 'Pavel Nadolsky',
  'authors': "C. Bal\\'azs, E. L. Berger, P. M. Nadolsky, C.-P. Yuan",
  'title': 'Calculation of prompt diphoton production cross sections at Tevatron and\n  LHC energies',
  'comments': '37 pages, 15 figures; published version',
  'journal-ref': 'Phys.Rev.D76:013009,2007',
  'doi': '10.1103/PhysRevD.76.013009',
  'report-no': 'ANL-HEP-PR-07-12',
  'categories': 'hep-ph',
  'license': None,
  'abstract': '  A fully differential calculation in perturbative quantum chromodynamics is\npresented for the production of massive photon pairs at hadron
colliders. All\nnext-to-leading order perturbative contributions from quark-antiquark,\ngluon-(anti)quark, and gluon-gluon subprocesses are included, as well
as\nall-orders resummation of initial-state gluon radiation valid at\nnext-to-next-to-leading logarithmic accuracy. The region of phase space is\nspecified
in which the calculation is most reliable. Good agreement is\ndemonstrated with data from the Fermilab Tevatron, and predictions are made for\nmore detailed
tests with CDF and DO data. Predictions are shown for\ndistributions of diphoton pairs produced at the energy of the Large Hadron\nCollider (LHC).
Distributions of the diphoton pairs from the decay of a Higgs\nboson are contrasted with those produced from QCD processes at the LHC, showing\nthat enhanced
sensitivity to the signal can be obtained with judicious\nselection of events.\n',
  'versions': [{'version': 'v1', 'created': 'Mon, 2 Apr 2007 19:18:42 GMT'},
   {'version': 'v2', 'created': 'Tue, 24 Jul 2007 20:10:27 GMT'}],
  'update_date': '2008-11-26',
  'authors_parsed': [['Balázs', 'C.', ''],
   ['Berger', 'E. L.', ''],
   ['Nadolsky', 'P. M.', ''],
   ['Yuan', 'C. -P.', '']]}]
```


Figure 2: Parsed json(sample)

2.1.2 Create a function that extracts and lists all fields (e.g., title, abstract, etc.) from the parsed RDD.

Here we us function *FieldExtractor* to impute missing values: (Note: It does'nt make *categories* field *N/A* for sake of reducing run-time of later steps)

```
def FieldExtractor(paper): # Also handles None values
        return {
        'id': paper.get('id') if paper.get('id') is not None else '
           N/A',
        .
        .Rest of fields
        .
        }
```

2.2 Preprocessing

In this section, we will clean the dataset by removing stop words and irrelevant characters to ensure the data is well-prepared for analysis.

2.2.1 Identify and remove or impute any null values, especially in critical fields

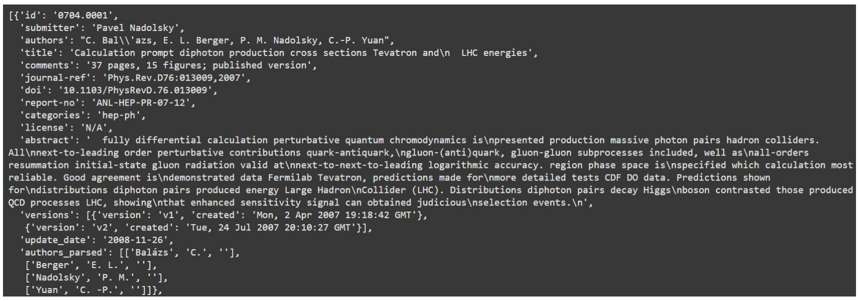
Already taken care of in previous section-part2:

FieldExtractor Function

2.2.2 Find and remove stopwords

we filer out stop-words via *StopWordsRemover* library which contains 150 stop-words.thereafter, Using function *process_stopwords*:

```
def process_stopwords(item):
 stopwords = set(StopWordsRemover.loadDefaultStopWords("english"))
 for stopword in stopwords:
 # ! only abstract and title fields have stopwords
  item['title'] = re.sub(fr' {stopword}$', ' ', item['title'], flags=re.
     IGNORECASE)
  item['title'] = re.sub(f' {stopword} ', ' ', item['title'], flags=re.
     IGNORECASE)
  item['abstract'] = re.sub(f' {stopword} ', ' ', item['abstract'], flags
     =re.IGNORECASE)
  item['abstract'] = re.sub(fr' {stopword}$', ' ', item['abstract'],
     flags=re.IGNORECASE)
return item
```



```
[{'id': '0704.0001',
  'submitter': 'Pavel Nadolsky',
  'authors': "C. Bal\\'azs, E. L. Berger, P. M. Nadolsky, C.-P. Yuan",
  'title': 'Calculation prompt diphoton production cross sections Tevatron and\n  LHC energies',
  'comments': '37 pages, 15 figures; published version',
  'journal-ref': 'Phys.Rev.D76:013009,2007',
  'doi': '10.1103/PhysRevD.76.013009',
  'report-no': 'ANL-HEP-PR-07-12',
  'categories': 'hep-ph',
  'license': 'N/A',
  'abstract': '  fully differential calculation perturbative quantum chromodynamics is\npresented production massive photon pairs hadron colliders.
All\nnext-to-leading order perturbative contributions quark-antiquark,\ngluon-(anti)quark, gluon-gluon subprocesses included, well as\nall-orders
resummation initial-state gluon radiation valid at\nnext-to-next-to-leading logarithmic accuracy. region phase space is\nspecified which calculation most
reliable. Good agreement is\ndemonstrated data Fermilab Tevatron, predictions made for\nmore detailed tests CDF DO data. Predictions shown
for\ndistributions diphoton pairs produced energy Large Hadron\nCollider (LHC). Distributions diphoton pairs decay Higgs\nboson contrasted those produced
QCD processes LHC, showing\nthat enhanced sensitivity signal can obtained judicious\nselection events.\n',
  'versions': [{'version': 'v1', 'created': 'Mon, 2 Apr 2007 19:18:42 GMT'},
   {'version': 'v2', 'created': 'Tue, 24 Jul 2007 20:10:27 GMT'}],
  'update_date': '2008-11-26',
  'authors_parsed': [['Balázs', 'C.', ''],
   ['Berger', 'E. L.', ''],
   ['Nadolsky', 'P. M.', ''],
   ['Yuan', 'C. -P.', '']]},
```


Figure 3: Filtering stop-words and None values

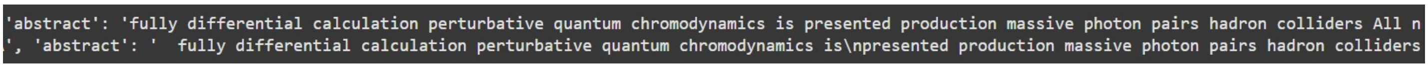
2.2.3 Find and remove useless characters

Using *cleaner* function we replace meaningless charachters with space:

Note that only *title* & *abstract* fields contain these charachters

```
def cleaner(page):
 replacements = {i: ' ' for i in chain(range(33, 64), range(123, 128),
    range(0, 32), range(91, 97))}
 replacements.update({ord('\n'): ' ',  ord('\u200c'): ' ', ord('\u200e'):
     ' ',
        ord('\u200d'): ' ', ord('¨C'): ' ', ord('_'): ' ', ord('?'): ' ',
            ord('.'): ' ',
```

```
        ord('?'): ' ',
})
page['title'] = re.sub(' {2,}', ' ', page['title'].translate(
    replacements)).strip()
page['abstract'] = re.sub(' {2,}', ' ', page['abstract'].translate(
    replacements)).strip()
return page
```

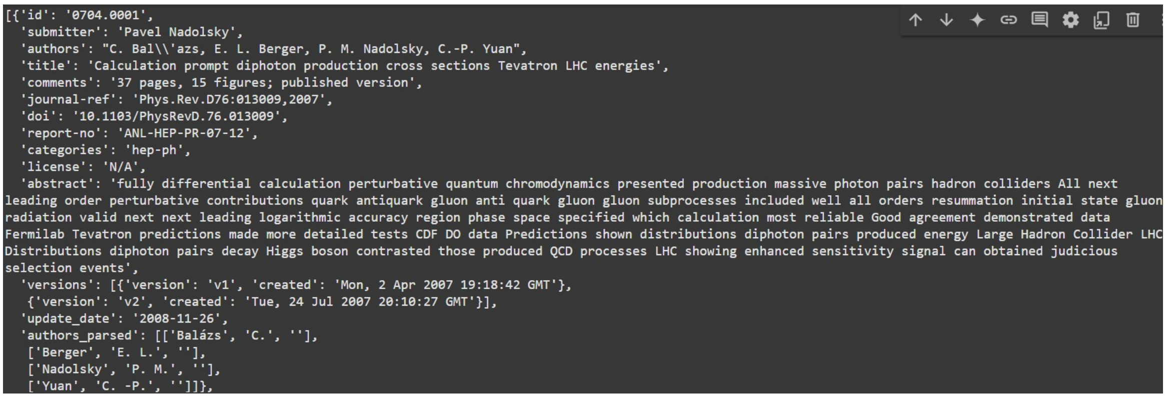


```
'abstract': 'fully differential calculation perturbative quantum chromodynamics is presented production massive photon pairs hadron colliders All n
', 'abstract': '  fully differential calculation perturbative quantum chromodynamics is\npresented production massive photon pairs hadron colliders
```

Figure 4: Sample with and without redundant characters

2.2.4 Even more purity!

Since after removing extra charachters, stopwords apear again,we need to remove the new stopwords which is done by running part 2.2.2




```
[{'id': '0704.0001',
  'submitter': 'Pavel Nadolsky',
  'authors': "C. Bal\\'azs, E. L. Berger, P. M. Nadolsky, C.-P. Yuan",
  'title': 'Calculation prompt diphoton production cross sections Tevatron LHC energies',
  'comments': '37 pages, 15 figures; published version',
  'journal-ref': 'Phys.Rev.D76:013009,2007',
  'doi': '10.1103/PhysRevD.76.013009',
  'report-no': 'ANL-HEP-PR-07-12',
  'categories': 'hep-ph',
  'license': 'N/A',
  'abstract': 'fully differential calculation perturbative quantum chromodynamics presented production massive photon pairs hadron colliders All next
leading order perturbative contributions quark antiquark gluon anti quark gluon gluon subprocesses included well all orders resummation initial state gluon
radiation valid next next leading logarithmic accuracy region phase space specified which calculation most reliable Good agreement demonstrated data
Fermilab Tevatron predictions made more detailed tests CDF DO data Predictions shown distributions diphoton pairs produced energy Large Hadron Collider LHC
Distributions diphoton pairs decay Higgs boson contrasted those produced QCD processes LHC showing enhanced sensitivity signal can obtained judicious
selection events',
  'versions': [{'version': 'v1', 'created': 'Mon, 2 Apr 2007 19:18:42 GMT'},
   {'version': 'v2', 'created': 'Tue, 24 Jul 2007 20:10:27 GMT'}],
  'update_date': '2008-11-26',
  'authors_parsed': [['Balázs', 'C.', ''],
   ['Berger', 'E. L.', ''],
   ['Nadolsky', 'P. M.', ''],
   ['Yuan', 'C. -P.', '']]},
```


Figure 5: Sample preprocessed element

2.3 Dataset Analysis

2.3.1 How many articles exist in each category (e.g., hep-ph, math.co)?

This snippet code will take care of the task:

```
                # Create (category, 1) pairs
fields_rdd.map(lambda record: (record['categories'], 1))
                # Aggregate counts
.reduceByKey(lambda a, b: a + b)
```

```
('math.NT math.AC', 119), ('math.AC math.GM math.NT math.RA', 1), ('math-ph cond-mat.str-el math.MP', 57), ('physics.acc-ph physics.optics', 145)
```

Figure 6: Sample *categories* count

2.3.2 Which category has the most articles?

Using *.map(lambda a: (a[1], a[0])).sortByKey()*:

```
[(86911, 'astro-ph'),
 (81999, 'hep-ph'),
 (71007, 'quant-ph'),
 (63257, 'cs.CV'),
 (59401, 'hep-th'),
 (40266, 'cond-mat.mtrl-sci'),
 (35492, 'cond-mat.mes-hall'),
 (35333, 'math.AP'),
 (31712, 'astro-ph.GA'),
 (31068, 'gr-qc')]
```

Figure 7: Highest rated *categories*

2.3.3 What is the distribution of the number of authors per article? (e.g., what percentage of articles have 1 author, more than 3 authors?)

Using comprehensive run of *.map(lambda x: len(x['authors'].split(','))).countByValue()* we get:

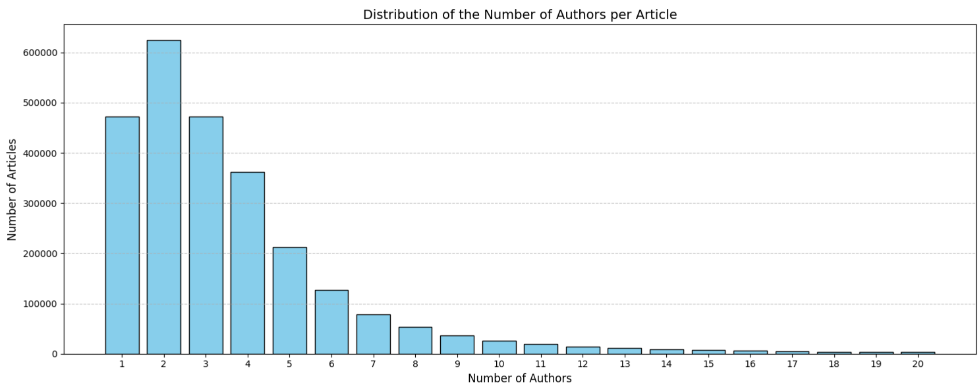


Figure 8: Highest # of *authors*

2.3.4 Filter out the articles that have more than 3 authors and generate a list of their titles and authors. Display the first 10 results.

- Filter(>3 collaborators):

.filter(lambda paper: len(paper['authors'].split(',')) > 3)

- map:

.map(lambda paper: (paper['title'], paper['authors']))

```
1. Title: Calculation prompt diphoton production cross sections Tevatron LHC energies, Authors: C. Bal\'azs, E. L. Berger, P. M. Nadolsky, C.-P. Yuan
2. Title: The Spitzer c d Survey Large Nearby Insterstellar Clouds IX Serpens YSO Population Observed IRAC MIPS, Authors: Paul Harvey, Bruno Merin, Tracy L. H
  Chapman, Neal J. Evans II, Philip C. Myers
3. Title: Lifetime doubly charmed baryons, Authors: Chao-Hsi Chang, Tong Li, Xue-Qian Li,Yu-Ming Wang
4. Title: Spectroscopic Observations Intermediate Polar EX Hydrae Quiescence, Authors: Nceba Mhlahlo, David H. Buckley, Vikram S. Dhillon, Steven B. Potter,
  Brian Warner,Patric A. Woudt
5. Title: Formation quasi solitons transverse confined ferromagnetic film media, Authors: A.A. Serga, M. Kostylev,,B. Hillebrands
6. Title: Spectroscopic Properties Polarons Strongly Correlated Systems Exact Diagrammatic Monte Carlo Method, Authors: A. S. Mishchenko (1,2),N. Nagaosa (1,3
  Science,Technology Agency, (2) Russian Research Centre ``Kurchatov
  Institute'', (3),University,Tokyo)
7. Title: Filling Factor Dependent Magnetophonon Resonance Graphene, Authors: M. O. Goerbig, J.-N. Fuchs, K. Kechedzhi, Vladimir I. Fal'ko
8. Title: Astrophysical gyrokinetics kinetic fluid turbulent cascades magnetized weakly collisional plasmas, Authors: A. A. Schekochihin (Oxford), S. C. Cowle
  (Maryland), G. W. Hammett (Princeton), G. G. Howes (Iowa), E. Quataert
  (Berkeley), T. Tatsuno (Maryland)
9. Title: Inference white dwarf binary systems using first round Mock LISA Data Challenges data sets, Authors: Alexander Stroeer, John Veitch, Christian Roeve
  Clark, Nelson Christensen, Martin Hendry, Chris Messenger, Renate Meyer,
  Matthew Pitkin, Jennifer Toher, Richard Umstaetter, Alberto Vecchio and
  Graham Woan
10. Title: Potassium intercalation graphite van der Waals density functional study, Authors: Eleni Ziambaras, Jesper Kleis, Elsebeth Schroder,,Per Hyldgaard
```

Figure 9: Articles with >3 *authors*

2.3.5 Plot a time series of the number of articles submitted per year.

Yet another comprehensive run by *.map(lambda x: (x['update_date'][:4], 1)).filter(lambda x: x[0] is not None).reduceByKey(lambda a, b: a + b).collect()* results:

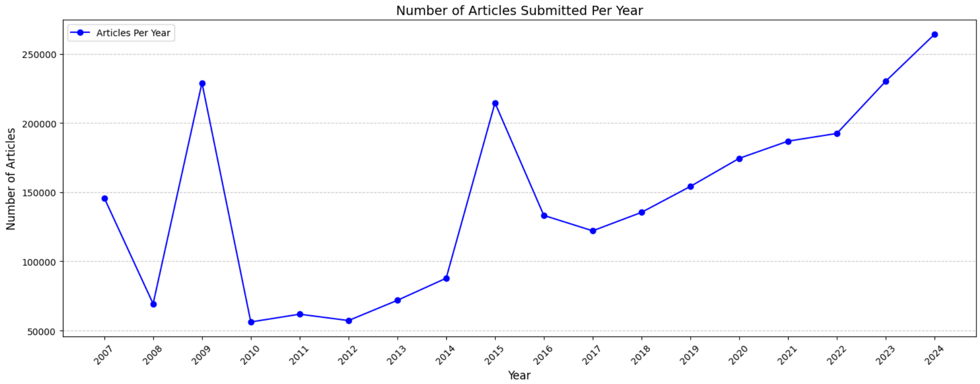


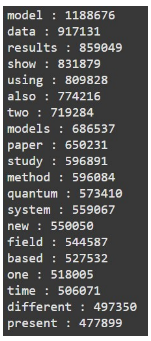
Figure 10: Number of articles per year

2.3.6 What are the 20 most frequent words in abstract? + 2.3.7: WordCloud

Using *.flatMap(process) & .map(lambda word: (word, 1)).reduceByKey(lambda a, b: a + b)* functions where

```python
def process(record):
 words = record['abstract'].lower().split()
 cleaned_words = [re.sub(r"[^a-zA-Z0-9]", "", word) for word in words]
return [word for word in cleaned_words if word and word not in stopwords]
```

we get:



```
model : 1188676
data : 917131
results : 859049
show : 831879
using : 809828
also : 774216
two : 719284
models : 686537
paper : 650231
study : 596891
method : 596084
quantum : 573410
system : 559067
new : 550050
field : 544587
based : 527532
one : 518005
time : 506071
different : 497350
present : 477899
```

Figure 11: Most frequent words in *abstract*

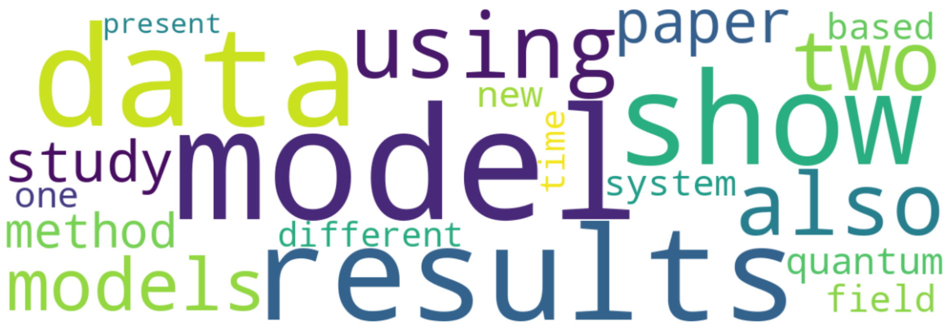


Figure 12: WordCloud

2.4 Advanced Data Exploration

• Due to comprehensive runtime of first 2 parts, we retrieved output only once in 3rd part.

Using:

- *.filter(lambda x: 'algorithm' in x['abstract'].re.IGNORECASE())*
- *.map(lambda x: (x['title'], len(x['abstract'].split())))*
- *.sortBy(lambda x: x[1], ascending=False).take(5)*

we get:

```
Title: The Nonlinearity Coefficient - A Practical Guide to Neural Architecture
  Design, Word Count: 498
Title: Generating a Generic Fluent API in Java, Word Count: 488
Title: Boxicity and Poset Dimension, Word Count: 484
Title: An Anytime Algorithm for Optimal Coalition Structure Generation, Word Count: 484
Title: McMini: A Programmable DPOR-Based Model Checker for Multithreaded
  Programs, Word Count: 475
```

Figure 13: Top 5 articles with the highest word counts in their *abstract* (containing 'algorithm')