**Problem Statement: To display a list of names, in decreasing order of probability of being the author for a given journal paper.**

We are using the Microsoft Academic Graph Dataset, which is a **27 GB** collection of all details of journal papers like author names, affiliations, keywords etc.

**Model 1**:
The first model will use keywords as the parameter for comparsion. We intend to generate a list of keywords used in all papers for each author. i.e. If author ABC has written 3 papers, the list will include keywords from all 3 papers.
Then, we get the keywords of the input paper, and find the most probable matches by comparing this list to all author-keyword lists.

**Pogress:**
For the first model, we need 2 files from the dataset -
**PaperAuthorAffiliations** (7GB) – which maps **paper_id** to **author_id**
**PaperKeywords** (5 GB)– which maps a list of **keyword_names** to **paper_id**

The first step was to find a way to process the large files. Importing it to R failed since it stores the data in memory. With 8 GB RAM, that wasn't an option.
JAVA Lucene is an API which indexes folders to make searches faster, but it took a long time (2-3 hours) to extract all distinct author_ids from the file.

Finally, the following logic was implemented in python to succesfully process the data-
1. The first part was to split the files into small manageable files of 10,000 lines each.

```
abhinav@ADell:/media/abhinav/Stuff/mag_paper$ ls -l MAG/PaperKeywords/splits| more
total 5.0G
drwxrwxrwx 1 root root 1.6M May  1 07:13 .
drwxrwxrwx 1 root root    0 Apr 30 19:42 ..
-rwxrwxrwx 1 root root 3.3M May  1 07:09 100000000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 06:57 10000000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 06:57 1000000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 06:57 100000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 100100000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 100200000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 100300000.PaperKeywords
-rwxrwxrwx 1 root root 3.1M May  1 07:09 100400000.PaperKeywords
-rwxrwxrwx 1 root root 3.2M May  1 07:09 100500000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 100600000.PaperKeywords
-rwxrwxrwx 1 root root 3.2M May  1 07:09 100700000.PaperKeywords
-rwxrwxrwx 1 root root 3.2M May  1 07:09 100800000.PaperKeywords
-rwxrwxrwx 1 root root 3.1M May  1 07:09 100900000.PaperKeywords
-rwxrwxrwx 1 root root 3.2M May  1 07:09 101000000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 06:57 10100000.PaperKeywords
-rwxrwxrwx 1 root root 3.2M May  1 07:09 101100000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 101200000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 101300000.PaperKeywords
-rwxrwxrwx 1 root root 3.2M May  1 07:09 101400000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 101500000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 101600000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 101700000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 101800000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 101900000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 102000000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 06:57 10200000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 102100000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 102200000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 102300000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 102400000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 102500000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 102600000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 102700000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 102800000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 102900000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 103000000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 06:57 10300000.PaperKeywords
-rwxrwxrwx 1 root root 3.3M May  1 07:09 103100000.PaperKeywords
 --More--
```

2. Next, we import the files one at a time into MongoDB

```
abhinav@ADell:~$ mongo
MongoDB shell version: 3.2.12
connecting to: test
Server has startup warnings:
2017-04-30T14:23:43.541+0530 I CONTROL  [initandlisten]
2017-04-30T14:23:43.541+0530 I CONTROL  [initandlisten] ** WARNING: /sys/kernel/mm/transparent_hugepage/enabled is 'always'.
2017-04-30T14:23:43.541+0530 I CONTROL  [initandlisten] **        We suggest setting it to 'never'
2017-04-30T14:23:43.541+0530 I CONTROL  [initandlisten]
> show dbs
local     0.000GB
lumongo   0.000GB
mag      28.727GB
> use mag
switched to db mag
> show collections
PaperAuthorAffiliations
PaperKeywords
testCollection
> db.PaperKeywords.find({"field_of_study" : "0623A455"}).pretty()
{
        "_id" : ObjectId("590692b33e094bc934ec6bc8"),
        "paper_id" : "091BFF3B",
        "keyword_name" : "elementary particles",
        "field_of_study" : "0623A455"
}
{
        "_id" : ObjectId("590692b33e094bc934ec6c61"),
        "paper_id" : "091EF8A9",
        "keyword_name" : "elementary particles",
        "field_of_study" : "0623A455"
}
{
        "_id" : ObjectId("590692b33e094bc934ec6d58"),
        "paper_id" : "0923402E",
        "keyword_name" : "elementary particles",
        "field_of_study" : "0623A455"
}
{
        "_id" : ObjectId("590692b33e094bc934ec6f1b"),
        "paper_id" : "092AB2B6",
        "keyword_name" : "elementary particles",
```

So now we have 2 collections for the 2 files. And the data can be queried as shown to get results quickly.

3. Finally, we use Pymongo (a python API for mongoDB) to generate a list of keywords for each author. This was done in 3 steps-

- First, we get a list of **distinct author_id** from the **PaperAuthorAffiliations** collection
- For **each author_id**, we get **all paper_id.**
- For **each paper_id** we get a list of **keywords** and append it to a file with the **filename as author_id**

So we now have a collection of files for each author, that contains a list of keywords used in their papers.