

# Paper Author Classification Problem

Vanjape Rajas Mangesh, Venkat Parthasarathy,  
Mahtab Sandhu

# The Problem

- **Input:-** Noisy dataset that contains academic papers and their probable authors.
- **Output:-** Infer actual authors from this set of probable authors.
- Problem arises due to:
  - Different authors having similar/identical names.
  - Authors may publish different papers using different names.

# Problem Challenges

- Huge Data
- Data Cleaning
- Missing data

# Dataset Information

- No. of Papers= >20 lakhs
- No. of authors= >3 lakhs
- **Features Available:-**
  - AuthorId
  - PaperId
  - JournalId where the paper was published
  - Keywords of the paper
  - Title of the paper
  - Affiliation of the author
- **Source of Dataset:-** Kaggle KDD Cup 2013

# Feature Explanation

1. 16 highest tf-idf of matching keywords of p and A.
2. 8 highest tf-idf scores of matching keywords of journal of p and A.
3. 4 highest number of overlapping “non-ambiguous” publications of A and those of other possible authors of p.
4. Number of keywords related to A.
5. Number of keywords occurring in p.
6. The maximum Levenshtein distance between the affiliation of the target author and affiliations of co authors in the papers
7. The minimum Levenshtein distance between affiliations of target author and co authors of paper

# Feature Extraction and Selection

- Create dictionaries for retrieving keywords of author, publications, and unambiguous publications of an author.
- To avoid creation of dictionaries every time, pickles were used. Generate these dictionaries only once, dump it using pickle and retrieve them whenever necessary.
- Minimize the number of times dictionaries are retrieved.
- For this interim, only the first five features mentioned in the paper were used.

# Validation Techniques

- As we have only implemented lazy learning classifiers (Naive Bayes and K-Means), no training was required.

# Performance Metrics

- **Mean Average Precision:-** Let the  $a_1..a_M$  be the list of publications actually written by the author  $a$ . Let  $b_1..b_N$  be the predicted order of the classifier. Then score is computed as follows:
  1. Let  $f(i)$  be no. of publications among  $b_1,...,b_i$  which are in list  $a_1,.. a_M$
  2. Then for every  $i$  when  $b_i$  is in the list  $a_1,..a_M$ ,  $f(i)/i$  is added to the score
  3. The score is divided by  $M$
- So, if all the actual publication are in the front of the list, the score will be equal to 1 while if they are elsewhere, the score will be low.



# K-means algorithm

- For a given author and a list of ambiguous publications, we find the features and run k-means algorithm onto it into cluster into two clusters.
- The publications under considered are sorted by the matching score with the keywords of author.
- The cluster having maximum average score is considered to be more probable and are placed before the other cluster in the probable order list.
- The matching score is computed as the sum of product of tf-idf scores of two words in the lists if they are similar.
- Two words are said to be similar if there levenshtein distance is less than 4

# Naive Bayes

- Assume that the probability density of the numerical features is normally distributed.
- Use maximum likelihood estimation to find the mean and standard deviation of the numerical features.
- Now, apply the Gaussian Distribution PD formula to find the probability of the numerical features and put them in Bayes formula.
- The list of publications are ordered in terms of the posterior probability. (Probability that publication  $p$  is written by author  $a$ )

# Support Vector Machine
















- The training data of svm is features corresponding the confirmed authors
- The svm is then trained using two kernels RBF and linear to predict whether the publication was written by that author
- Then the result is sorted according to levenshtein distances

# Random Forest and Decision Trees

- They were implemented using python libraries
- Output is similar to SVM

# Results for the classifiers implemented

- K-Means Classifier

65	 1488	PRIS_803 	<a href="#">0.84046</a>	9	<a href="#">Tue, 25 Jun 2013 09:04:47 (-26.2h)</a>
66	 1319	Nilesh 	<a href="#">0.84046</a>	3	<a href="#">Wed, 26 Jun 2013 19:21:43</a>
67	 1186	Sherlock_Holmes 	<a href="#">0.79267</a>	15	<a href="#">Tue, 25 Jun 2013 15:17:54</a>
68	 1326	Baser	<a href="#">0.74682</a>	5	<a href="#">Wed, 26 Jun 2013 09:24:56</a>
69	 1397	deserttosnow 	<a href="#">0.72109</a>	24	<a href="#">Wed, 26 Jun 2013 22:24:36</a>
-		<b>vmrajas</b>	<b>0.70994</b>	-	<b>Fri, 30 Sep 2016 19:48:09</b> <small>Post-Deadline</small>
<b>Post-Deadline Entry</b> If you would have submitted this entry during the competition, you would have been around here on the leaderboard.					
70	 1330	LM 	<a href="#">0.65489</a>	3	<a href="#">Fri, 21 Jun 2013 14:48:36</a>
	Valid.csv Benchmark		0.00000		
	Random Benchmark		0.00000		
	Basic Coauthor Benchmark		0.00000		
	Basic Python Benchmark		0.00000		

# Results for the classifiers implemented

- Naive Bayes Classifier

65	↑488	PRIS_803 🏠	0.84046	9	Tue, 25 Jun 2013 09:04:47 (-26.2h)
66	↑319	Nilesh 🏠	0.84046	3	Wed, 26 Jun 2013 19:21:43
67	↑186	Sherlock_Holmes 🏠	0.79267	15	Tue, 25 Jun 2013 15:17:54
68	↑326	Baser	0.74682	5	Wed, 26 Jun 2013 09:24:56
69	↑397	deserttosnow 🏠	0.72109	24	Wed, 26 Jun 2013 22:24:36
-		<b>VenkatParthasarathy</b>	<b>0.65575</b>	-	<b>Fri, 30 Sep 2016 19:54:30</b> <b>Post-Deadline</b>
<b>Post-Deadline Entry</b> If you would have submitted this entry during the competition, you would have been around here on the leaderboard.					
70	↑330	LM 🏠	0.65489	3	Fri, 21 Jun 2013 14:48:36
📍	Valid.csv Benchmark		0.00000		
📍	Random Benchmark		0.00000		
📍	Basic Coauthor Benchmark		0.00000		
📍	Basic Python Benchmark		0.00000		

# Results

Upon choosing a random sample of size 100 from the test data and verifying with the output of the paper, the results are shown below:

Classifier	Mean Average Precision
Naive Bayes	0.78
K-Means	0.85
SVM	0.82
Decision Tree	0.83
Random Forest	0.86

# Instructions

## SMAI-Project

---

### Prerequisites:-

---

1. Python3
2. Pickle
3. Numpy
4. Pandas

### Installation:-

---

1. Run

```
git clone https://github.com/venkatp1997/SMAI-Project.git
```

2. Create a directory "dataRev2" inside the repo.
3. Download the [data](#) and unzip it inside dataRev2.
4. Run

```
python init.py
```

from the root directory of the repo to create the pickles.

5. Run

```
python main.py 0
```

for Naive Bayes Classifier or

```
python main.py 1
```

for K-Means Classifier.