

DS5230 Unsupervised Machine Learning

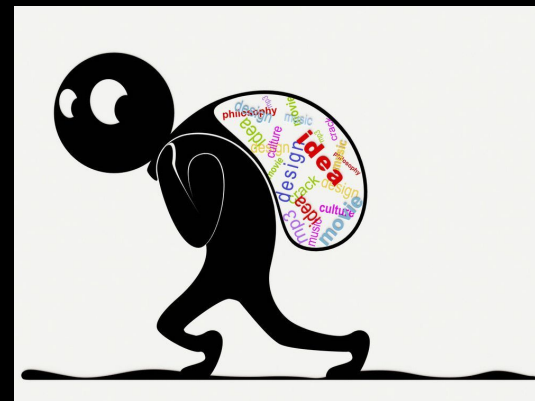
Plagiarism Detection

Team Members:

Aishwarya Vantipuli,

Spatika Krishnan,

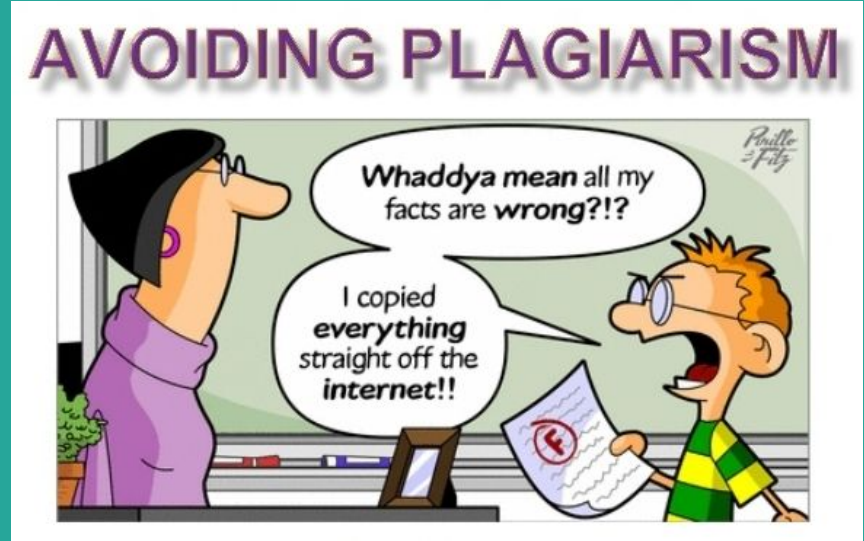
Nandini Jampala



Plagiarism is the wrongful appropriation and stealing and publication of another author's language, thoughts, ideas, or expressions and the representation of them as one's own original work.

Business Application:

- Verifying Originality
- Tracking Content Misuse
- In-Depth Plagiarism Analysis



Dataset

Publicly available dataset
Corpus of Plagiarised Short Answers was chosen.

| | File | task | Text |
|---|----------------|------|---|
| 0 | g0pA_taska.txt | a | inheritance is a basic concept of object orien... |
| 1 | g0pA_taskb.txt | b | pagerank is a link analysis algorithm used by ... |
| 2 | g0pA_taskc.txt | c | the vector space model also called term vector... |
| 3 | g0pA_taskd.txt | d | bayes theorem was names after rev thomas bayes... |
| 4 | g0pA_taske.txt | e | dynamic programming is an algorithm design tec... |
| 5 | g0pB_taska.txt | a | inheritance is a basic concept in object orien... |
| 6 | g0pB_taskb.txt | b | pagerank pr refers to both the concept and the... |
| 7 | g0pB_taskc.txt | c | vector space model is an algebraic model for r... |
| 8 | g0pB_taskd.txt | d | bayes theorem relates the conditional and marg... |

Student id_task id

Question name

Uncleaned Text

The corpus consists of answers to the following five short questions on a variety of topics in the Computer Science field:

- What is inheritance in object-oriented programming?
- Explain the PageRank algorithm that is used by the Google search engine.
- Explain the Vector Space Model that is used for Information Retrieval
- Explain Bayes Theorem from probability theory.
- What is dynamic programming?

Each question has 19 students answers and 1 wikipedia answer

Dataset:

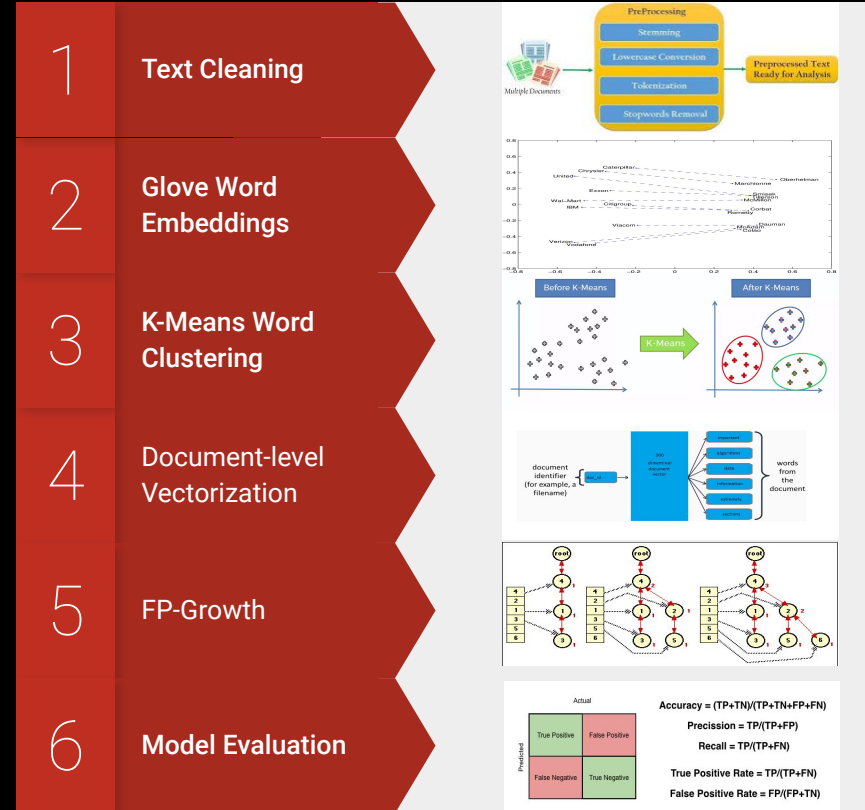
https://ir.shef.ac.uk/cloughie/resources/plagiarism_corpus.html

Method 1

Goal of the Project:

The goal of this project is to find the list of students with plagiarised work for from the student answers given as input.

For this we proposed two methods.



Method 1

Text Pre-Processing:

- All punctuations and stop words are removed and sentences are tokenized using NLTK

Glove Embeddings:

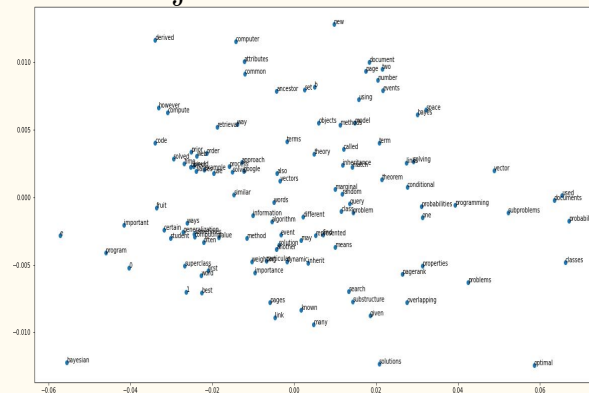
List of words in each sentence

```
['inheritance', 'basic', 'concept', 'objectoriented', 'programming', 'basic', 'idea', 'create', 'new', 'classes', 'add', 'extra', 'detail', 'exist', 'done', 'allowing', 'new', 'classes', 'reuse', 'methods', 'variables', 'existing', 'classes', 'new', 'methods', 'classes', 'added', 'specialise', 'inheritance', 'models', 'isindof', 'relationship', 'entities', 'objects', 'example', 'postgraduates', 'undergraduates', 'kinds', 'student']  
['break', 'problem', 'different', 'smaller', 'subproblems']  
['pagerank', 'derived', 'theoretical', 'probability', 'value', 'logarithmic', 'scale', 'like', 'richter', 'scale']  
['richard', 'bellman', 'originally', 'coined', 'term', '1940s', 'describe', 'method', 'solving', 'problems', 'one', 'needs', 'find', 'best', 'dec
```

GLOVE



PCA Projection of Word Embeddings



K-Means Clustering:

- Using the K-means algorithm, we cluster these word vectors into K clusters based on their contextual meaning. The output of this step is K word clusters.

Method 1

Document level Vectorization:

Sentence vector: each entry is a word representing cluster label it belongs to.

```
taska {'g0pA': ([0, 1, 2, 0, 1, 1, 2, 4, 4, 5, 2, 5, 5, 4], [2, 2, 4, 4, 5, 4, 2, 5, 4, 4, 4, 4, 2, 2, 4, 6], [0, 7, 7, 2, 2, 6, 5, 2, 2, 2,
#####
taskb {'g0pA': ([10, 5, 9, 0, 9, 2, 1, 2, 2, 2, 1, 2, 2, 1, 12, 7, 7, 10, 2, 2, 2, 2, 2, 1], [9, 2, 2, 1, 2, 2, 10], [2, 2, 2, 9], [10],
#####
taskc {'g0pB': ([0, 0, 16, 0, 16, 0, 0, 5, 5, 17, 18, 18, 17], [2, 2, 16, 2, 4], [16, 0, 0, 0, 0, 5, 6, 16, 0], [6, 6, 2, 5, 0, 2, 5, 16,
#####
taskd {'g0pA': ([22, 22, 0, 7, 9, 0, 22, 0, 12, 0, 0, 1, 7, 12, 1, 6, 1, 7, 1], [0, 24, 24, 1, 0, 1, 24, 24], [22, 0], [0, 12, 22], [2, 7, 7,
#####
taskd {'g0pB': ([0, 0], [5, 1, 0, 9, 0, 0, 21], [2], [21], [0, 0], [6, 2, 12, 7, 7, 0], [0, 0, 0], [24], [0, 0], [0], [0, 9], [1]), 'g0pA':
```

Types of questions in corpus.

Student id

Document Vector:
each entry is sentence
numbered according to
the rule.

Rule:

We form answer vector by assigning a unique number if two sentence vectors are different. If two sentence vectors are same we assign each of them the same number.

```
taska {'g0pA': [0, 1, 2, 3, 4, 5, 6, 7], 'g0pB': [8, 9, 10, 11, 920, 13, 1075, 15, 16, 17, 18, 19], 'g0pC': [20, 21, 22, 1032, 24, 1067, 26,
####
taskb {'g0pA': [390, 227, 228, 424, 230, 424, 232, 233, 391, 235, 938, 409, 410, 239, 1052], 'g0pB': [241, 242, 243, 244, 245, 246, 247], 'g0pC': [248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000], 'g0pD': [1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 10
```

FP-GROWTH

| Transactions | Itemset (Min sup > 3) |
|--------------|---|
| g0pB_taske | [686, 876, 875, 794, 879, 880] |
| g0pB_taske | [241, 242, 243, 244, 245, 246, 247] |
| g0pA_taska | [0, 1, 2, 3, 4, 5, 6, 7] |
| g0pB_taska | [8, 9, 10, 11, 920, 13, 1075, 15, 16, 17, 18, 19] |
| g1pB_taskc | [489, 631, 632, 634, 635, 636, 1052] |
| | |
| | |
| | |

100 Documents



| Question | List of students with plagiarism |
|----------|--|
| Task e | { 'g1pB', 'orig', 'g3pB', 'g0pE', 'g4pB' } |
| Task a | { 'g3pC', 'orig', 'g2pC', 'g0pE', 'g4pC' } |
| Task d | { 'g2pA', 'g2pB', 'g0pB', 'orig', 'g3pA', 'g0pC', 'g1pA' } |

Method-1

Model Evaluation:

- With Binary Classification, i.e, Heavily Plagiarised and Non Plagiarised the model yielded pretty good results.
- With Multi-Class Classification i.e, light, heavy, cut, non Plagiarised levels, Precision is high. This is mainly because of imbalanced classes

Binary Classification

| | |
|-----------|--------|
| Accuracy | 79.03% |
| Precision | 90% |
| Recall | 53.55% |
| F1-Score | 68.81% |

Multi-Class Classification

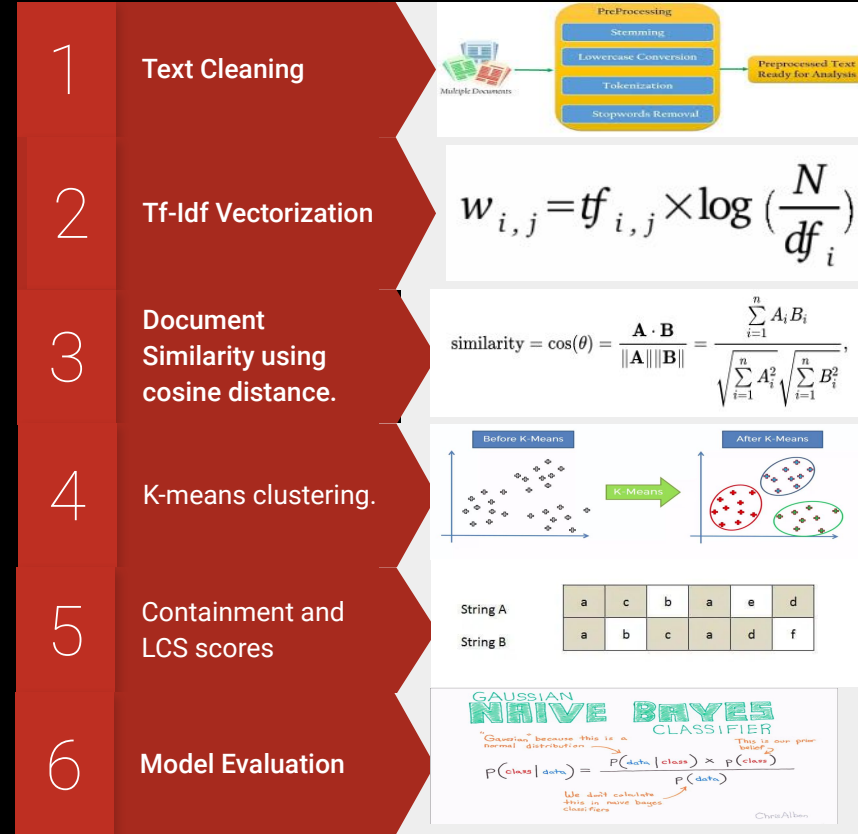
| | |
|-----------|--------|
| Accuracy | 61% |
| Precision | 75% |
| Recall | 45.5% |
| F1-Score | 66.23% |

Goal of the Project:

To find out the list of students who have plagiarised the tasks and to what extent.

Second Method consists of following steps...>

Method 2



Tf-idf Vectorization

- First count word occurrences by document.
- Then apply the term frequency-inverse document frequency weighting.

$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$

| | Doc 1 | Doc 2 | ... | Doc n |
|-----------|-------|-------|-----|-------|
| Term(s) 1 | 12 | 2 | ... | 1 |
| Term(s) 2 | 0 | 1 | ... | 0 |
| ... | ... | ... | ... | ... |
| Term(s) n | 0 | 6 | ... | 3 |

Tf-idf Output:

Vector form of g0pA_taska document

```
[ [ 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0.57501852 0.34113499 0. 0. 0.
  0. 0. 0. 0.47152341 0.57501852 0. ]
 [ 0. 0. 0. 0.40824829 0.40824829 0.
  0.40824829 0. 0. 0. 0. 0.
  0.40824829 0.40824829 0.40824829 0. 0.
  0. 0.490779 0.490779 0. 0. 0.
  0. 0. 0. 0.490779 0.4024458 0.
  0.26396385 0. 0. 0. 0.52792769
  0. 0.79189154 0. 0. 0.15659897 0.
  0. 0. 0. 0. 0. 0.
  0. 0.59430676 0. 0.35257792 0.59430676
  0. 0. 0. 0. 0.41144595 ]
 [ 0. 0. 0. 0. 0. 0.
  0. 0.59430676 0. 0.35257792 0.59430676
  0. 0. 0. 0. 0.41144595 ]
 [ 0. 0. 0. 0.26990725 0. 0.9099135
  0. 0. 0. 0. 0.
  0.31497221 ] ]
```

- Words that occur frequently within a document but not frequently within the corpus receive a higher weighting as these words are assumed to contain more meaning in relation to the document

Cosine similarity:

- Cosine similarity is measured against the tf-idf matrix. It returns a 100*100 matrix that contains the distance measure of each document with every other document in our corpus.

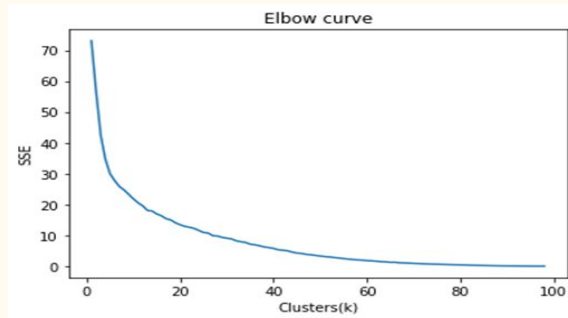
Output of Cosine Similarity

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 1 |
|---|-----------|-----------|-----------|----------|----------|----------|----------|----------|----------|----------|-----|---|
| 0 | 1.000000 | 1.000000 | -0.001518 | 0.029564 | 0.029697 | 0.317527 | 0.000018 | 0.000018 | 0.000018 | 0.000018 | ... | - |
| 1 | 1.000000 | 1.000000 | -0.001518 | 0.029564 | 0.029697 | 0.317527 | 0.000018 | 0.000018 | 0.000018 | 0.000018 | ... | - |
| 2 | -0.001518 | -0.001518 | 1.000000 | 0.040489 | 0.031902 | 0.042664 | 0.000094 | 0.000094 | 0.000094 | 0.000094 | ... | C |
| 3 | 0.029564 | 0.029564 | 0.040489 | 1.000000 | 0.999642 | 0.081541 | 0.017496 | 0.017496 | 0.017496 | 0.017496 | ... | C |
| 4 | 0.029697 | 0.029697 | 0.031902 | 0.999642 | 1.000000 | 0.077855 | 0.017598 | 0.017598 | 0.017598 | 0.017598 | ... | C |
| 5 | 0.317527 | 0.317527 | 0.042664 | 0.081541 | 0.077855 | 1.000000 | 0.122909 | 0.122909 | 0.122909 | 0.122909 | ... | - |
| 6 | 0.000018 | 0.000018 | 0.000094 | 0.017496 | 0.017598 | 0.122909 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | ... | C |
| 7 | 0.000018 | 0.000018 | 0.000094 | 0.017496 | 0.017598 | 0.122909 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | ... | C |
| 8 | 0.000018 | 0.000018 | 0.000094 | 0.017496 | 0.017598 | 0.122909 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | ... | C |
| 9 | 0.000018 | 0.000018 | 0.000094 | 0.017496 | 0.017598 | 0.122909 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | ... | C |

K-means clustering

- Using the cosine similarity matrix, you can run a slew of clustering algorithms to better understand the hidden structure within the synopses.

Hyperparameter Selection Method for No. of Cluster (K)



Here the elbow curves at $k=8$ and hence our predetermined number of clusters is 8

Output of K-means clustering

| title | cluster |
|------------|---------|
| gOpA_taska | 0 |
| gOpA_taskb | 4 |
| gOpA_taskc | 3 |
| gOpA_taskd | 1 |
| gOpA_taske | 6 |
| gOpB_taska | 6 |
| gOpB_taskb | 7 |
| gOpB_taskc | 3 |
| gOpB_taskd | 1 |
| gOpB_taske | 0 |
| gOpC_taska | 0 |
| gOpC_taskb | 5 |
| gOpC_taskc | 3 |
| gOpC_taskd | 1 |
| gOpC_taske | 0 |
| gOpD_taska | 0 |
| gOpD_taskb | 4 |
| gOpD_taskc | 3 |
| gOpD_taskd | 1 |
| gOpD_taske | 0 |
| gOpE_taska | 6 |
| gOpE_taskb | 4 |
| gOpE_taskc | 3 |
| gOpE_taskd | 1 |

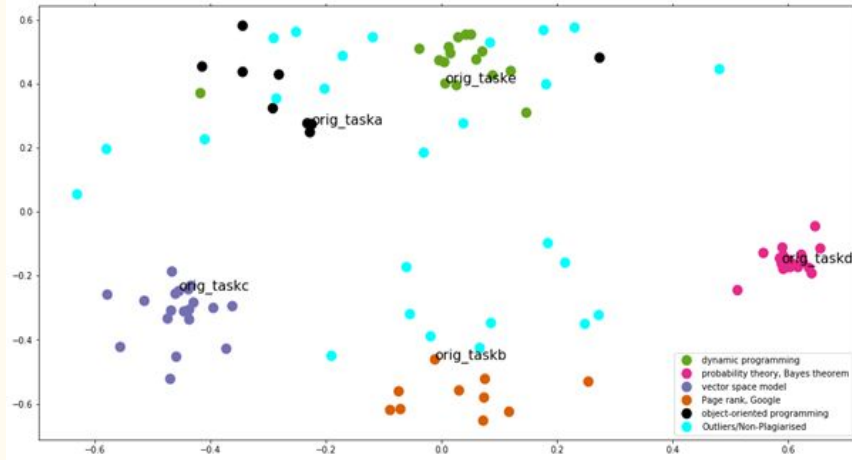
Each Students answer is assigned a cluster label from 0-7.

The original(wikipedia answers) is also assigned a cluster.



| | |
|------------|---|
| orig_taska | 6 |
| orig_taskb | 4 |
| orig_taskc | 3 |
| orig_taskd | 1 |
| orig_taske | 0 |

PCA to visualize document clusters

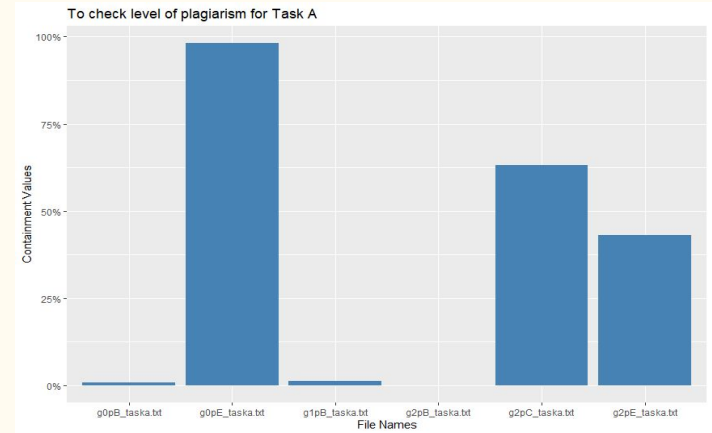


Now to classify the documents into the levels of plagiarism, we calculate the containment and LCS scores between the original document and the documents that share the same cluster as that.

Feature Extraction:

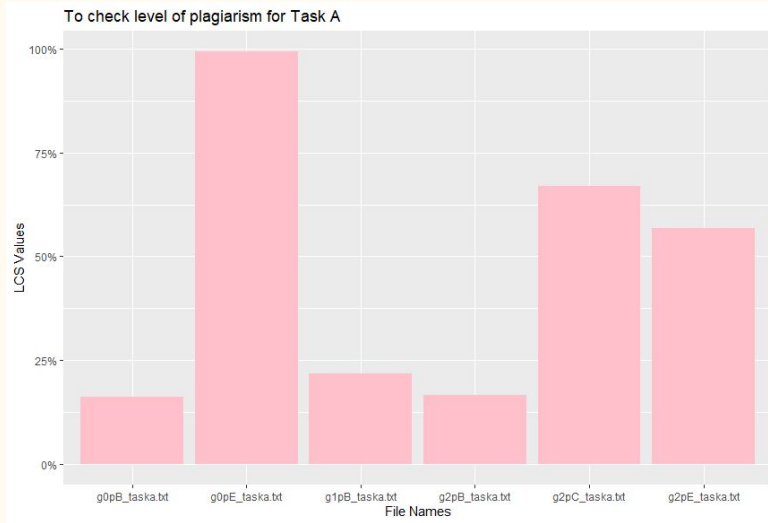
Containment:

For the specified student task, we compared it's n-gram count to the corresponding wikipedia n-gram count to calculate the containment values.



Least Common Subsequence:

The longest common subsequence is the longest string of words that are the same between the wikipedia task text and the student answer.



Classification Metrics :

| | |
|-----------|-----|
| Accuracy | 92% |
| Precision | 93% |
| Recall | 92% |
| F1-Score | 92% |

Conclusion

- The accuracy of text models depends heavily on the preprocessing methods of the text data.
- In Method 1, using FP-Growth, we observed a high precision and accuracy scores of 90% and 79% respectively for Binary Classification.
- This behaviour fell as multi-level classification is implementation with a precision of 75%. This is mainly due to Class Imbalances.
- In Method 2, we observed a high precision as well as high accuracy scores of 93% and 92% respectively.
- We've experimented with k-means clustering and hierarchical clustering in both methods. K-means give better results as the other is not suitable for large datasets and Initial seeds have a strong impact on the final results thus giving us inaccurate cluster labels.