In [4]:
```
# Day 1 AM Lab 1
import numpy as np

# this returns a number whose probability of occurence is p
def sampleValue (p):
        return np.flatnonzero (np.random.multinomial (1, p, 1))[0]

# there are 2000 words in the corpus
alpha = np.full (2000, .1)

# there are 100 topics
beta = np.full (100, .1)

# this gets us the probabilty of each word happening in each of the 100 topics
wordsInTopic = np.random.dirichlet (alpha, 100)
# wordsInCorpus[i] will give us the number of each word in the document
wordsInCorpus = {}

# generate each doc
for doc in range (0, 50):
        #
        # no words in this doc yet
        wordsInDoc = {}
        #
        # get the topic probabilities for this doc
        topicsInDoc = np.random.dirichlet (beta)
        #
        # generate each of the 1000 words in this document
        for word in range (0, 1000):
                #
                # select the topic and the word
                whichTopic = sampleValue (topicsInDoc)
                whichWord = sampleValue (wordsInTopic[whichTopic])
                #
                # and record the word
                wordsInDoc [whichWord] = wordsInDoc.get (whichWord, 0) + 1
                #
        # now, remember this document
        wordsInCorpus [doc] = wordsInDoc
print(wordsInCorpus [doc])
print(type(wordsInCorpus [doc]))
print(wordsInCorpus[4])
```

{948: 1, 4: 1, 314: 1, 759: 2, 940: 6, 662: 2, 403: 4, 1158: 1, 1133: 3, 235:
2, 833: 3, 275: 2, 149: 4, 1519: 1, 780: 3, 1450: 3, 1571: 2, 608: 1, 1846:
2, 792: 4, 1674: 1, 1948: 2, 1712: 1, 186: 3, 1270: 2, 1076: 1, 1328: 2, 103:
1, 1645: 1, 460: 1, 747: 4, 336: 5, 546: 3, 622: 1, 1324: 6, 1982: 3, 951: 1,
1575: 2, 173: 1, 63: 1, 632: 1, 1344: 1, 1247: 2, 389: 1, 492: 1, 797: 3, 102
6: 2, 495: 1, 16: 1, 1492: 2, 713: 2, 1719: 3, 847: 2, 771: 1, 1877: 1, 1109:
1, 328: 3, 1940: 4, 1318: 2, 938: 1, 1642: 3, 738: 2, 1280: 2, 1548: 1, 412:
3, 763: 2, 48: 1, 1303: 1, 536: 1, 520: 2, 534: 1, 1668: 2, 402: 4, 376: 2, 1
98: 1, 1618: 1, 1847: 5, 1339: 4, 6: 1, 1395: 1, 1099: 3, 84: 2, 1741: 1, 13
0: 1, 372: 1, 92: 3, 486: 1, 1515: 4, 1470: 1, 592: 2, 1915: 4, 1034: 4, 143
0: 2, 1894: 1, 979: 2, 1841: 3, 724: 2, 1513: 4, 392: 1, 609: 2, 382: 2, 106
8: 2, 497: 1, 1934: 2, 1021: 7, 465: 1, 1496: 6, 627: 1, 1072: 1, 1458: 1, 19
12: 1, 1156: 2, 1404: 3, 1511: 1, 1279: 2, 1985: 1, 1202: 1, 697: 2, 825: 2,
1965: 8, 1208: 1, 363: 2, 1800: 1, 956: 1, 141: 1, 778: 2, 748: 2, 1810: 1, 8
2: 1, 522: 1, 170: 1, 1091: 2, 822: 6, 1916: 1, 1919: 2, 1152: 2, 166: 1, 178
4: 2, 292: 4, 218: 1, 1082: 2, 1677: 2, 1307: 4, 263: 3, 1250: 4, 1057: 1, 10
97: 2, 423: 3, 971: 1, 25: 1, 1421: 2, 195: 3, 711: 2, 75: 2, 1077: 1, 1522:
1, 518: 1, 61: 2, 1512: 1, 1231: 1, 1336: 1, 1132: 1, 572: 1, 1935: 1, 2: 1,
864: 3, 433: 2, 3: 1, 1997: 1, 1273: 1, 1644: 3, 1887: 1, 369: 2, 942: 1, 163
2: 1, 1272: 2, 1830: 1, 194: 2, 145: 1, 1474: 1, 253: 2, 1434: 3, 251: 1, 187
9: 2, 1818: 1, 1295: 1, 99: 1, 214: 2, 997: 1, 645: 1, 682: 3, 903: 5, 1956:
1, 42: 2, 1449: 1, 762: 1, 213: 2, 1521: 2, 1267: 5, 321: 3, 231: 1, 1346: 2,
1024: 1, 823: 1, 1285: 1, 872: 1, 1950: 1, 1765: 2, 1315: 1, 1478: 2, 649: 3,
1563: 4, 1281: 1, 1179: 1, 552: 1, 683: 1, 222: 1, 274: 3, 450: 1, 36: 2, 141
8: 3, 1958: 1, 1981: 1, 519: 4, 302: 1, 659: 1, 1523: 3, 22: 3, 1277: 2, 665:
3, 717: 2, 1353: 1, 1213: 1, 1802: 1, 613: 1, 320: 2, 467: 1, 1192: 1, 1237:
2, 1367: 1, 967: 1, 1118: 1, 1888: 3, 1085: 3, 774: 1, 1524: 1, 1770: 1, 889:
1, 343: 1, 1257: 1, 1266: 1, 1961: 3, 28: 1, 1978: 1, 1838: 2, 379: 1, 574:
1, 564: 1, 87: 1, 902: 1, 1123: 1, 703: 1, 1669: 1, 1261: 1, 1001: 1, 267: 1,
50: 1, 1046: 1, 223: 3, 898: 2, 1790: 2, 1365: 1, 1175: 1, 1855: 3, 931: 2, 1
236: 1, 225: 2, 1215: 1, 72: 1, 930: 1, 1429: 1, 790: 2, 1227: 5, 1150: 2, 76
4: 1, 1178: 1, 1604: 2, 304: 1, 1480: 7, 565: 1, 783: 1, 920: 1, 331: 1, 126
3: 1, 153: 1, 1914: 1, 1100: 1, 1210: 1, 211: 4, 1746: 1, 351: 2, 1054: 1, 17
61: 1, 341: 1, 1845: 1, 49: 2, 135: 4, 7: 1, 323: 1, 1151: 1, 371: 1, 1652:
1, 45: 3, 830: 3, 106: 1, 1704: 2, 1676: 1, 1058: 1, 1069: 2, 1964: 2, 14: 1,
248: 1, 1471: 1, 1209: 2, 1525: 1, 765: 1, 134: 2, 358: 1, 1736: 1, 511: 1, 1
494: 1, 1071: 1, 557: 1, 1684: 3, 882: 1, 162: 3, 996: 1, 234: 3, 436: 2, 58
6: 1, 669: 1, 1420: 1, 993: 1, 1756: 4, 1374: 1, 285: 2, 1999: 2, 846: 2, 25
5: 2, 1984: 1, 1304: 4, 161: 2, 1589: 1, 281: 4, 70: 1, 739: 3, 447: 1, 702:
2, 1196: 1, 1243: 1, 458: 2, 322: 3, 1322: 1, 147: 1, 98: 1, 1357: 1, 475: 2,
1509: 1, 360: 1, 1293: 1, 1401: 1, 1890: 3, 104: 1, 1444: 1, 1702: 1, 229: 1,
238: 3, 297: 2, 69: 1, 139: 2, 880: 1, 900: 2, 1611: 2, 1568: 1, 1898: 1, 138
3: 1, 1753: 1, 1432: 1, 461: 1, 110: 1, 1706: 1, 1861: 1, 1710: 1, 966: 1, 14
23: 1, 983: 1, 62: 1, 721: 1, 1486: 2, 1061: 1, 1737: 1, 172: 2, 1053: 1, 178
1: 1, 120: 1, 1014: 1, 631: 1, 171: 1, 1619: 2, 658: 2, 1011: 1, 1772: 1, 41
0: 1, 1869: 2, 1913: 1, 442: 1, 1448: 1, 1465: 1, 1597: 1, 568: 1, 361: 2, 11
91: 1, 1457: 1, 338: 1, 1650: 1, 1708: 1, 1447: 1, 695: 1, 859: 1, 1758: 2, 8
77: 1, 1414: 1, 1809: 1, 165: 1, 1943: 1, 667: 1, 837: 1, 80: 1, 1717: 1, 45
5: 1, 768: 1, 1673: 1, 1893: 1, 329: 1, 416: 1, 1895: 2, 573: 1, 1647: 2, 69
0: 1, 675: 1, 501: 1, 174: 1, 1558: 1, 1292: 1, 746: 1, 1526: 1, 1507: 3, 198
7: 1, 1464: 3, 1081: 1, 1438: 1, 105: 1, 51: 1, 468: 1, 1019: 1, 1074: 1, 28
4: 1, 1856: 1, 954: 1, 1593: 3, 1008: 1, 1699: 1, 1942: 1, 1095: 1, 1372: 2,
1427: 1, 1228: 1, 393: 1, 330: 2, 1296: 1, 434: 2, 474: 1, 1147: 2, 876: 1, 3
53: 2, 1218: 1, 350: 1, 184: 2, 1366: 1, 59: 2, 895: 1, 1456: 1, 798: 1, 606:
1, 291: 1, 1268: 1, 652: 1, 1388: 1, 1460: 1, 1373: 1, 1787: 2, 1271: 1, 199
0: 1, 1106: 2, 505: 1, 1591: 1, 316: 1, 556: 1, 169: 1, 1063: 1, 1135: 1, 111
2: 1, 432: 2, 1412: 2, 1234: 1, 1517: 1, 634: 1, 125: 3, 258: 1, 638: 1, 164

```
3: 1, 1976: 2, 1127: 2, 1104: 2, 743: 2, 60: 1, 679: 1, 791: 4, 1726: 1, 643:
1, 516: 1, 691: 1, 535: 1, 129: 1, 1967: 1, 221: 1, 1925: 1, 226: 1, 1685: 1,
888: 1, 1543: 1, 1043: 1, 677: 1, 1164: 1, 672: 1, 1088: 1, 1749: 1, 939: 2,
1798: 1, 975: 1, 370: 1, 354: 1, 530: 1, 1108: 1, 1282: 1, 1451: 1, 394: 1, 9
64: 1, 1214: 1, 543: 2, 55: 1, 355: 1, 340: 1, 325: 1, 1462: 1, 1305: 1, 245:
1, 646: 1, 1974: 1, 901: 1, 1313: 1, 1168: 1, 15: 1, 112: 1, 1572: 1, 1153:
1, 236: 1, 840: 1, 114: 1, 1145: 1, 1216: 1, 1601: 1, 1723: 1, 1932: 1, 933:
1, 601: 1, 949: 1, 470: 1, 582: 1, 342: 1, 65: 1, 839: 1, 588: 1, 637: 1, 190
9: 1, 508: 1, 457: 1, 35: 2, 1835: 1, 563: 1, 414: 1, 480: 1, 1416: 1, 1078:
1, 1600: 1, 860: 1, 1825: 1, 1639: 3, 452: 1, 1413: 1, 1204: 1, 1323: 1, 193
0: 1, 585: 1, 1394: 1, 308: 1, 261: 1, 1574: 1, 362: 1, 540: 1, 1738: 1, 367:
1, 1472: 1, 1567: 1, 1621: 1, 906: 1, 628: 1, 1125: 1, 66: 1, 1782: 1, 359:
1, 375: 1, 44: 1, 1240: 1, 1351: 1, 205: 1, 1217: 1, 469: 1, 1387: 1, 1253:
1, 925: 1, 720: 1, 327: 1, 1436: 1, 38: 1, 499: 1, 1461: 1, 1221: 1, 927: 1,
1637: 1, 249: 1, 1183: 1, 709: 1, 897: 1, 1051: 1}
<class 'dict'>
{606: 2, 1465: 1, 1952: 2, 694: 3, 1379: 1, 1536: 1, 106: 1, 984: 2, 1625: 2,
872: 2, 1773: 1, 1315: 1, 1975: 1, 748: 2, 1104: 4, 1640: 1, 608: 2, 1455: 1,
5: 1, 387: 1, 1463: 1, 41: 1, 1790: 2, 1808: 1, 832: 2, 244: 1, 796: 2, 454:
1, 592: 8, 957: 2, 1958: 3, 230: 2, 1352: 1, 104: 2, 486: 1, 638: 1, 1785: 1,
1593: 1, 540: 2, 1726: 2, 1066: 2, 1478: 2, 758: 2, 1106: 5, 1924: 1, 1943:
1, 1326: 2, 621: 1, 475: 3, 57: 2, 1439: 1, 1774: 1, 530: 5, 174: 2, 443: 1,
653: 3, 1353: 2, 1140: 3, 743: 6, 76: 2, 399: 1, 1566: 2, 633: 1, 1322: 2, 11
01: 1, 1856: 1, 1585: 1, 980: 1, 1624: 1, 436: 1, 1148: 1, 664: 1, 132: 3, 73
0: 1, 1069: 1, 1243: 1, 1154: 1, 216: 3, 1040: 3, 1491: 1, 426: 2, 761: 3, 10
39: 3, 1448: 1, 1639: 4, 44: 1, 622: 3, 1832: 1, 1916: 2, 1233: 1, 685: 1, 13
40: 2, 1076: 3, 1622: 1, 1962: 4, 31: 3, 675: 2, 1988: 2, 159: 1, 652: 3, 77
9: 1, 905: 1, 1026: 3, 1163: 2, 1275: 1, 1345: 1, 225: 1, 277: 5, 21: 4, 401:
2, 1300: 1, 357: 1, 896: 1, 1671: 1, 989: 2, 1783: 1, 734: 1, 587: 1, 1436:
2, 1772: 1, 403: 2, 1865: 1, 969: 1, 347: 2, 396: 1, 963: 1, 469: 1, 1834: 1,
1583: 1, 1740: 2, 1327: 2, 1869: 1, 715: 2, 411: 1, 1248: 1, 598: 1, 1514: 1,
1013: 2, 1057: 1, 1369: 1, 1472: 2, 1051: 1, 907: 1, 1891: 3, 188: 1, 964: 2,
875: 1, 169: 1, 1195: 3, 1277: 1, 459: 1, 1144: 2, 1349: 3, 1286: 1, 1087: 1,
141: 1, 1004: 1, 235: 3, 625: 1, 1494: 2, 1397: 2, 105: 1, 861: 1, 1722: 1, 1
549: 2, 1760: 1, 280: 2, 950: 2, 500: 1, 17: 2, 1170: 3, 1953: 1, 1395: 1, 43
9: 3, 1526: 2, 860: 3, 36: 3, 1214: 1, 177: 1, 1390: 2, 686: 1, 1119: 1, 168
4: 3, 1902: 1, 356: 1, 441: 2, 979: 2, 1273: 1, 615: 1, 1767: 2, 1964: 2, 33
6: 1, 1383: 1, 164: 2, 507: 1, 1311: 1, 915: 1, 183: 2, 114: 1, 770: 1, 1483:
1, 597: 2, 1393: 2, 1361: 1, 700: 1, 48: 1, 1200: 1, 689: 2, 15: 2, 899: 1, 1
607: 2, 1307: 1, 760: 3, 434: 2, 414: 1, 845: 1, 1480: 3, 1152: 2, 1699: 1, 7
37: 2, 432: 1, 163: 2, 825: 3, 956: 1, 326: 1, 524: 1, 658: 2, 222: 1, 798:
1, 505: 3, 543: 2, 1616: 1, 1255: 3, 1099: 1, 959: 1, 499: 1, 1343: 1, 1228:
1, 541: 1, 1387: 4, 1520: 1, 847: 2, 236: 1, 1117: 1, 1173: 1, 1843: 1, 1380:
3, 1725: 1, 1371: 3, 886: 1, 1259: 1, 1816: 2, 1289: 2, 808: 2, 8: 2, 377: 1,
1657: 1, 1751: 1, 1704: 1, 285: 2, 265: 2, 1434: 1, 1820: 1, 435: 1, 1643: 1,
497: 1, 1528: 1, 1682: 3, 1201: 2, 39: 3, 172: 2, 1094: 1, 626: 1, 1100: 2, 9
31: 2, 1088: 1, 26: 1, 1199: 1, 623: 1, 805: 1, 946: 3, 1606: 2, 827: 1, 732:
2, 471: 1, 120: 1, 1604: 2, 752: 1, 1331: 1, 176: 1, 305: 3, 522: 3, 670: 1,
962: 1, 362: 3, 1806: 1, 1015: 1, 301: 4, 1636: 1, 1744: 1, 35: 1, 903: 4, 7
3: 2, 604: 3, 1573: 1, 954: 1, 228: 1, 1782: 1, 1809: 2, 714: 2, 201: 2, 155
1: 5, 1678: 2, 790: 2, 171: 2, 665: 1, 1972: 3, 1650: 1, 1085: 2, 1044: 1, 85
8: 2, 636: 1, 1923: 1, 930: 1, 1909: 1, 1611: 2, 1373: 2, 11: 3, 422: 3, 379:
1, 1648: 1, 1710: 1, 532: 1, 80: 1, 1337: 1, 209: 1, 1274: 1, 644: 1, 656: 1,
1589: 2, 1559: 1, 1928: 1, 257: 2, 313: 1, 193: 1, 421: 2, 1299: 1, 358: 2, 1
61: 3, 1996: 2, 1859: 2, 1619: 1, 662: 1, 1623: 2, 196: 2, 319: 2, 1453: 1, 1
541: 3, 1729: 1, 45: 2, 331: 1, 690: 1, 603: 1, 942: 2, 1092: 1, 1319: 2, 184
0: 1, 1305: 1, 97: 1, 40: 1, 992: 2, 703: 2, 1486: 2, 820: 1, 1391: 2, 1985:
```

1, 1766: 1, 1218: 2, 1991: 1, 683: 1, 1939: 1, 629: 2, 1579: 1, 129: 1, 1368:
1, 468: 1, 437: 2, 287: 2, 290: 2, 1162: 1, 1333: 1, 1207: 3, 828: 1, 1535:
1, 1407: 1, 856: 1, 766: 1, 1020: 1, 986: 2, 116: 1, 1250: 1, 1123: 1, 149:
2, 27: 1, 204: 1, 877: 1, 294: 1, 1567: 1, 307: 1, 618: 1, 1059: 2, 1021: 1,
781: 1, 574: 1, 1881: 1, 1109: 1, 1656: 1, 717: 1, 639: 1, 1279: 1, 1481: 1,
687: 1, 1080: 2, 392: 1, 757: 1, 43: 1, 332: 1, 1543: 1, 445: 2, 274: 1, 156
0: 1, 309: 1, 863: 1, 674: 1, 765: 1, 1212: 2, 1966: 1, 1814: 1, 342: 2, 119
3: 1, 709: 1, 93: 1, 744: 2, 61: 1, 1482: 1, 1883: 3, 1221: 1, 855: 2, 271:
1, 1743: 1, 1441: 1, 1595: 1, 1889: 2, 1907: 2, 699: 1, 1796: 1, 1577: 1, 118
2: 1, 237: 2, 1817: 1, 1378: 1, 1955: 1, 203: 2, 318: 1, 1893: 1, 982: 1, 199
8: 1, 390: 1, 1959: 1, 1934: 1, 736: 2, 971: 2, 1365: 1, 848: 1, 727: 1, 185:
1, 4: 1, 1635: 1, 1690: 1, 842: 1, 413: 1, 1603: 1, 1858: 1, 344: 1, 1563: 1,
1768: 3, 602: 1, 1927: 1, 561: 1, 1761: 1, 1323: 1, 369: 2, 1204: 1, 549: 1,
906: 1, 1438: 1, 1723: 1, 1071: 2, 472: 1, 1632: 2, 1911: 2, 1406: 1, 1125:
1, 1970: 1, 1797: 1, 1940: 1, 575: 1, 1153: 1, 619: 1, 1339: 1, 386: 1, 383:
1, 1857: 1, 1799: 1, 751: 1, 65: 1, 788: 1, 846: 1, 672: 1, 1285: 2, 1244: 1,
844: 1, 526: 1, 100: 1, 1320: 1, 69: 1, 1325: 1, 1295: 2, 1265: 1, 1569: 1, 1
496: 2, 578: 1, 1565: 2, 933: 1, 1561: 1, 807: 2, 628: 1, 1495: 1, 577: 2, 13
03: 1, 1147: 1, 1868: 1, 1512: 4, 1098: 1, 329: 1, 1341: 1, 1202: 1, 517: 1,
1457: 1, 911: 2, 1389: 1, 503: 1, 154: 1, 372: 1, 1394: 1, 701: 1, 918: 1, 24
2: 1, 1360: 1, 1518: 1, 916: 2, 523: 1, 1186: 1, 510: 1, 302: 1, 777: 1, 142
3: 1, 355: 1, 254: 1, 144: 1, 1546: 1, 1036: 1, 1941: 1, 780: 1, 2: 1, 705:
1, 350: 1, 278: 1, 571: 1, 1452: 1, 1659: 1, 1458: 2, 1830: 1, 1711: 2, 1227:
2, 298: 2, 1355: 1, 487: 1, 1780: 1, 506: 1, 1879: 1, 92: 1, 997: 1, 1309: 1,
1115: 2, 375: 2, 970: 1, 1914: 1, 733: 2, 1896: 1, 1904: 1, 1719: 1, 451: 1,
1910: 1, 1166: 1, 345: 1, 772: 1, 1613: 1, 566: 1, 1223: 1, 1366: 1, 691: 1,
908: 1, 1983: 1, 295: 1, 1885: 1, 563: 1, 1167: 4, 546: 2, 600: 1, 537: 1, 16
44: 1, 655: 1, 1701: 1, 24: 1, 1308: 1, 720: 1, 1460: 2, 75: 1, 857: 1, 511:
1, 1005: 1, 1476: 1, 673: 1, 1677: 1, 1588: 1, 198: 1, 1412: 1, 764: 1, 755:
1, 1446: 2, 1467: 1, 1114: 2, 266: 1, 1679: 1, 1297: 1, 1459: 1, 1733: 1, 107
7: 2, 1454: 1, 624: 1, 86: 1, 1329: 1, 756: 1, 1028: 1, 1866: 1, 1781: 1, 158
1: 1, 64: 1, 410: 1, 1987: 1, 1658: 1, 1150: 1, 233: 1, 818: 1, 465: 1, 904:
1, 822: 1, 901: 1, 125: 1, 1401: 1, 1715: 1, 1976: 2, 50: 1, 671: 1, 374: 1,
1116: 1, 215: 1, 1875: 1, 883: 2, 354: 1, 1358: 1, 1489: 1, 240: 1, 1511: 1,
692: 1, 1956: 1, 178: 1, 1198: 1, 1498: 1, 110: 1, 433: 1, 483: 1, 416: 1, 51
2: 1, 1413: 1, 1851: 1, 1485: 1}

In [10]:
```
# Day 1 AM Lab 2
import numpy as np

# there are 2000 words in the corpus
alpha = np.full (2000, .1)

# there are 100 topics
beta = np.full (100, .1)

# this gets us the probabilty of each word happening in each of the 100 topics
wordsInTopic = np.random.dirichlet (alpha, 100)

# wordsInCorpus[i] will give us the vector of words in document i
wordsInCorpus = np.zeros ((50, 2000))

# generate each doc
for doc in range (0, 50):
        #
        # get the topic probabilities for this doc
        topicsInDoc = np.random.dirichlet (beta)
        #
        # assign each of the 1000 words in this doc to a topic
        wordsToTopic = np.random.multinomial (1000, topicsInDoc)
        #
        # and generate each of the 1000 words
        for topic in range (0, 100):
                wordsFromCurrentTopic = np.random.multinomial (wordsToTopic[to
pic], wordsInTopic[topic])
                wordsInCorpus[doc] = np.add (wordsInCorpus[doc], wordsFromCurr
entTopic)
print(wordsInCorpus [doc])
print(type(wordsInCorpus [doc]))
print(wordsInCorpus[2,1:10])
```

```
[2. 0. 0. ... 0. 2. 0.]
<class 'numpy.ndarray'>
[1. 1. 0. 0. 0. 0. 0. 1. 0.]
```

In [13]:
```
# Day 1 AM Lab 2 - alternate code by teacher
import numpy as np

# there are 2000 words in the corpus
alpha = np.full (2000, .1)

# there are 100 topics
beta = np.full (100, .1)

# this gets us the probabilty of each word happening in each of the 100 topics
wordsInTopic = np.random.dirichlet (alpha, 100)

# wordsInCorpus[i] will give us the vector of words in document i
wordsInCorpus = np.zeros ((50, 2000))

# generate each doc
for doc in range (0, 50):
        #
        # get the topic probabilities for this doc
        topicsInDoc = np.random.dirichlet (beta)
        #
        # assign each of the 1000 words in this doc to a topic
        wordsFromCurrentTopic = wordsToTopic[topic]
        #
        # and generate each of the 1000 words
        for topic in range (0, 100):
                wordsFromCurrentTopic = wordsToTopic[topic]
                wordsInCorpus[doc] = np.add (wordsInCorpus[doc], np.random.mul
tinomial (wordsToTopic[topic], wordsInTopic[topic]))
print(wordsInCorpus [doc])
print(type(wordsInCorpus [doc]))
print(wordsInCorpus[2,1:10])
```

```
[0. 0. 0. ... 1. 0. 0.]
<class 'numpy.ndarray'>
[1. 0. 1. 0. 3. 1. 1. 0. 0.]
```

In [11]:
```
np.zeros ((50, 2000))
```

Out[11]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

In [ ]:
```
# Day 1 AM Lab 3
import numpy as np
# there are 2000 words in the corpus
alpha = np.full (2000, .1)

# there are 100 topics
beta = np.full (100, .1)

# this gets us the probabilty of each word happening in each of the 100 topics
wordsInTopic = np.random.dirichlet (alpha, 100)

# produced [doc, topic, word] gives us the number of times that the given word
was
# produced by the given topic in the given doc
produced = np.zeros ((50, 100, 2000))

# generate each doc
for doc in range (0, 50):
        #
        # get the topic probabilities for this doc
        topicsInDoc = np.random.dirichlet (beta)
        #
        # assign each of the 1000 words in this doc to a topic
        wordsToTopic = np.random.multinomial (1000, topicsInDoc)
        #
        # and generate each of the 1000 words
        for topic in range (0, 100):
                produced[doc, topic] = np.random.multinomial (wordsToTopic[top
ic], wordsInTopic[topic])

#
```

In [15]:
```
# 1 Write a line of code that computes the number of words produced by topic 1
7 in document 18.
produced[18,17,:].sum () # or produced[18,17].sum ()
```

Out[15]: 0.0

In [16]:
```
# 1 Write a line of code that computes the number of words produced by topic 1
7 in document 18.
produced[18,17].sum ()
```

Out[16]: 0.0

In [20]:
```
# 2 Write a line of code that computes the number of words produced by topic 1
7 thru 45 in document 18.
produced[18,17:46].sum ()
```

Out[20]: 668.0

In [22]:
```
# 3 Write a line of code that computes the number of words in the entire corpus. = 50 docs * 1000 words to topic
produced.sum () # or produced[:,:,:].sum ()
```

Out[22]: 50000.0

In [23]:
```
# 4 Write a line of code that computes the number of words in the entire corpus produced by topic 17 .
produced[:,17,:].sum () # or produced[:,17].sum ()
```

Out[23]: 204.0

In [24]:
```
# 4 Write a line of code that computes the number of words in the entire corpus produced by topic 17 .
produced[:,17].sum ()
```

Out[24]: 204.0

In [25]:
```
# 5 Write a line of code that computes the number of words in the entire corpus produced by topic 17 or topic 23.
produced[:,np.array([17,23]),:].sum ()
```

Out[25]: 598.0

In [46]:
```
# 5 Write a line of code that computes the number of words in the entire corpus produced by topic 17 or topic 23.
#  NOT correct version uses addition
produced[:,17].sum () + produced[:,23].sum ()
```

Out[46]: 598.0

In [47]:
```
# 6 Write a line of code that computes the number of words in the entire corpus produced by even numbered topics.
produced[:,np.arange(0,100,2),:].sum ()
```

Out[47]: 23468.0

In [48]:
```
# ODD numbered topics
produced[:,np.arange(1,100,2),:].sum ()
```

Out[48]: 26532.0

In [49]:
```
# 7 Write a line of code that computes the number of each word produced by topic 15.
produced[:,15,:].sum (0) # or produced.sum (0)[15]
```

Out[49]: array([0., 0., 0., ..., 0., 0., 3.])

In [50]:
```
# 7 Write a line of code that computes the number of each word produced by topic 15.
produced.sum (0)[15]
```

Out[50]: array([0., 0., 0., ..., 0., 0., 3.])

In [51]:
```
# 8 Write a line of code that computes the topic responsible for the most instances of each word in the corpus.
produced.sum (0).argmax (0)
```

Out[51]:
```
array([ 8, 74, 20, ..., 83, 35, 69], dtype=int64)
```

In [45]:
```
# 9 Write a line of code that for each topic, computes the max number of occurences (summed over all documents) of any word that it was responsible for.
produced[:,np.arange(0,100,1),produced.sum (0).argmax (1)].sum(0) # This works, though it is possible to come up with a much better solution!
```

Out[45]:
```
array([19., 16.,  8., 17., 20., 14., 10., 23., 13., 17., 15.,  7., 10.,
        8., 10., 16.,  7.,  9., 15., 12., 11., 23., 17., 10., 13., 13.,
       15., 14., 12., 15.,  8., 17., 12., 20., 24., 17.,  7.,  8.,  4.,
       13., 18., 17., 15.,  9., 14.,  7., 14.,  7.,  8., 15.,  7., 31.,
        9., 15.,  5., 12., 12.,  7., 10., 10., 16., 11., 13., 16., 21.,
        9.,  8.,  8.,  9., 12., 11.,  7., 18., 12., 11., 25., 13., 21.,
        9., 16., 15., 23., 21., 25.,  8., 12., 14., 12., 11., 23., 17.,
       10.,  8., 14., 12., 10., 16., 14.,  5., 12.])
```

In [55]:
```
# 9 Write a line of code that for each topic, computes the max number of occurences (summed over all documents) of any word that it was responsible for.
produced[:,np.arange(0,100),produced.sum (0).argmax (1)].sum(0)
```

Out[55]:
```
array([19., 16.,  8., 17., 20., 14., 10., 23., 13., 17., 15.,  7., 10.,
        8., 10., 16.,  7.,  9., 15., 12., 11., 23., 17., 10., 13., 13.,
       15., 14., 12., 15.,  8., 17., 12., 20., 24., 17.,  7.,  8.,  4.,
       13., 18., 17., 15.,  9., 14.,  7., 14.,  7.,  8., 15.,  7., 31.,
        9., 15.,  5., 12., 12.,  7., 10., 10., 16., 11., 13., 16., 21.,
        9.,  8.,  8.,  9., 12., 11.,  7., 18., 12., 11., 25., 13., 21.,
        9., 16., 15., 23., 21., 25.,  8., 12., 14., 12., 11., 23., 17.,
       10.,  8., 14., 12., 10., 16., 14.,  5., 12.])
```

In [5]:
```
# Day 1 AM Lab 4
import numpy as np
# initialize data
import numpy as np

# this returns a number whose probability of occurence is p
def sampleValue (p):
        return np.flatnonzero (np.random.multinomial (1, p, 1))[0]

# there are 2000 words in the corpus
alpha = np.full (2000, .1)

# there are 100 topics
beta = np.full (100, .1)

# this gets us the probabilty of each word happening in each of the 100 topics
wordsInTopic = np.random.dirichlet (alpha, 100)
# wordsInCorpus[i] will give us the number of each word in the document
wordsInCorpus = {}

# generate each doc
for doc in range (0, 50):
        #
        # no words in this doc yet
        wordsInDoc = {}
        #
        # get the topic probabilities for this doc
        topicsInDoc = np.random.dirichlet (beta)
        #
        # generate each of the 1000 words in this document
        for word in range (0, 1000):
                #
                # select the topci and the word
                whichTopic = sampleValue (topicsInDoc)
                whichWord = sampleValue (wordsInTopic[whichTopic])
                #
                # and record the word
                wordsInDoc [whichWord] = wordsInDoc.get (whichWord, 0) + 1
                #
        # now, remember this document
        wordsInCorpus [doc] = wordsInDoc
print(wordsInCorpus [doc])
print(type(wordsInCorpus [doc]))
```

{116: 1, 25: 1, 210: 2, 1717: 3, 393: 1, 783: 1, 255: 3, 1331: 1, 285: 1, 148
6: 4, 1870: 2, 566: 2, 1523: 4, 127: 2, 1176: 2, 1990: 2, 1225: 1, 1715: 2, 1
056: 2, 553: 1, 1175: 2, 834: 1, 1825: 1, 1078: 1, 188: 1, 916: 2, 1804: 2, 9
26: 3, 800: 2, 1405: 4, 1974: 2, 1443: 2, 1616: 1, 1491: 1, 1865: 2, 649: 2,
523: 1, 1271: 2, 54: 1, 63: 4, 1409: 2, 50: 2, 604: 2, 365: 2, 667: 1, 62: 1,
555: 2, 159: 1, 1114: 1, 1552: 1, 21: 1, 245: 3, 504: 1, 1171: 2, 788: 1, 195
8: 3, 753: 2, 1519: 3, 513: 1, 1460: 1, 619: 1, 1627: 1, 1734: 1, 618: 1, 150
3: 1, 1385: 2, 314: 1, 719: 2, 747: 1, 1138: 1, 994: 2, 456: 1, 1600: 1, 134
8: 3, 570: 1, 999: 2, 564: 2, 782: 1, 430: 1, 760: 1, 425: 1, 1966: 2, 1686:
2, 1874: 4, 1178: 4, 1553: 1, 1793: 1, 1961: 2, 738: 1, 854: 1, 438: 1, 1536:
2, 1916: 2, 1388: 3, 602: 1, 422: 2, 769: 2, 793: 2, 1585: 2, 1861: 1, 961:
2, 1068: 3, 1010: 3, 761: 3, 700: 1, 1706: 1, 857: 1, 606: 1, 235: 6, 1306:
3, 678: 1, 1336: 1, 1209: 1, 1746: 1, 1226: 1, 278: 3, 1872: 1, 119: 2, 226:
1, 1650: 2, 1518: 2, 1906: 1, 806: 1, 1528: 5, 1139: 1, 746: 3, 1069: 3, 108
3: 1, 176: 2, 462: 1, 1153: 1, 996: 1, 271: 1, 1658: 1, 1054: 1, 1512: 1, 45
7: 2, 1031: 1, 650: 3, 1562: 1, 428: 1, 1457: 2, 721: 1, 1935: 1, 1196: 1, 19
39: 2, 195: 1, 394: 4, 801: 1, 512: 1, 722: 2, 269: 1, 1185: 1, 398: 9, 1119:
1, 599: 2, 345: 1, 1169: 1, 779: 4, 1285: 2, 7: 1, 515: 1, 1346: 1, 1305: 1,
6: 1, 503: 1, 1963: 1, 339: 1, 636: 1, 1328: 1, 1292: 3, 581: 2, 1505: 1, 106
7: 2, 1559: 1, 536: 1, 397: 1, 1258: 3, 501: 2, 644: 4, 1444: 1, 279: 1, 29:
1, 1158: 1, 799: 1, 744: 1, 1456: 1, 1411: 1, 1332: 1, 525: 3, 387: 2, 803:
1, 1816: 1, 371: 1, 79: 1, 946: 1, 1291: 1, 1752: 1, 569: 1, 486: 2, 1159: 3,
1802: 1, 1890: 1, 1827: 2, 304: 1, 357: 2, 1988: 1, 1452: 1, 1207: 2, 1402:
1, 87: 3, 237: 1, 794: 1, 1812: 1, 1899: 3, 1266: 2, 593: 1, 743: 1, 879: 1,
1670: 2, 95: 2, 284: 3, 1489: 2, 1473: 1, 686: 1, 626: 4, 1123: 3, 1846: 1, 1
121: 2, 441: 2, 1572: 4, 383: 1, 1387: 1, 1704: 1, 1889: 3, 287: 2, 1241: 2,
1851: 1, 1281: 3, 623: 2, 1614: 3, 919: 1, 1739: 1, 386: 2, 532: 1, 988: 1, 1
490: 1, 1222: 1, 1563: 1, 137: 4, 317: 1, 412: 1, 1784: 1, 1702: 1, 549: 8, 1
278: 1, 789: 1, 302: 1, 1719: 1, 967: 3, 254: 1, 1118: 1, 1290: 1, 1437: 2, 8
15: 3, 717: 1, 727: 1, 1978: 1, 177: 1, 160: 1, 1726: 2, 1345: 2, 1395: 1, 17
98: 2, 930: 1, 507: 1, 375: 1, 1878: 3, 651: 2, 1810: 2, 578: 1, 1876: 2, 162
8: 1, 647: 1, 1151: 1, 1555: 2, 178: 2, 1223: 1, 1938: 3, 945: 2, 1862: 1, 10
21: 2, 1154: 1, 121: 2, 929: 1, 1383: 2, 1830: 1, 140: 1, 1301: 1, 1214: 2, 9
71: 1, 337: 1, 1172: 1, 1205: 1, 1921: 1, 1430: 1, 1741: 1, 612: 1, 775: 1, 2
41: 1, 1560: 1, 1446: 2, 790: 3, 729: 1, 98: 1, 1905: 1, 559: 2, 1046: 1, 38
4: 1, 171: 1, 1805: 1, 481: 1, 1756: 1, 1406: 1, 318: 1, 104: 1, 724: 2, 784:
1, 1724: 1, 1866: 2, 1262: 1, 997: 1, 249: 1, 145: 1, 1509: 2, 1408: 1, 1626:
2, 565: 1, 933: 2, 601: 1, 286: 1, 1808: 3, 80: 1, 567: 1, 1573: 2, 154: 1, 1
31: 1, 466: 1, 1424: 2, 1504: 1, 673: 1, 1954: 2, 687: 1, 1357: 1, 1124: 1, 1
777: 1, 1427: 2, 963: 1, 1125: 4, 1111: 1, 884: 1, 713: 1, 936: 1, 1143: 1, 1
803: 1, 1160: 1, 1340: 1, 410: 1, 1221: 3, 628: 1, 1762: 4, 1482: 2, 426: 1,
182: 1, 1651: 2, 1206: 1, 1188: 2, 415: 1, 202: 1, 1967: 1, 125: 1, 465: 1, 8
49: 1, 1989: 1, 1740: 1, 995: 3, 1052: 2, 19: 1, 1472: 1, 1366: 1, 732: 1, 97
8: 1, 752: 2, 1997: 2, 1421: 1, 1608: 3, 795: 1, 1696: 1, 955: 1, 1677: 1, 16
46: 2, 1367: 1, 1429: 1, 10: 2, 526: 1, 1902: 1, 92: 1, 1753: 1, 1534: 1, 147
8: 3, 1293: 1, 179: 2, 36: 1, 1732: 1, 33: 2, 86: 1, 543: 1, 927: 2, 1510: 1,
458: 1, 1601: 1, 1965: 1, 1933: 2, 1096: 3, 206: 1, 1718: 1, 1058: 2, 231: 2,
1951: 2, 1026: 1, 11: 2, 911: 1, 1000: 1, 1252: 1, 1228: 1, 1914: 1, 1014: 1,
1242: 1, 1061: 2, 902: 3, 1979: 1, 258: 1, 1326: 1, 712: 1, 595: 1, 1763: 1,
1279: 1, 14: 1, 648: 2, 1465: 1, 1132: 1, 870: 2, 449: 1, 611: 1, 259: 1, 86
9: 1, 1020: 1, 920: 1, 1950: 1, 409: 2, 417: 1, 444: 1, 484: 1, 13: 2, 1177:
1, 585: 2, 720: 1, 1936: 2, 896: 1, 1960: 2, 1722: 1, 238: 2, 126: 1, 1379:
1, 1152: 1, 1590: 1, 824: 2, 5: 1, 1829: 1, 1401: 1, 298: 1, 491: 1, 1880: 1,
47: 1, 807: 1, 739: 1, 1828: 1, 1075: 1, 1312: 3, 1439: 1, 1038: 1, 993: 1, 9
17: 2, 1857: 1, 1157: 1, 1024: 1, 891: 1, 205: 2, 592: 1, 937: 1, 730: 1, 32
5: 1, 485: 1, 861: 2, 69: 2, 222: 1, 261: 2, 459: 2, 1926: 1, 1901: 1, 1076:
1, 787: 2, 27: 1, 1944: 1, 1836: 1, 1776: 1, 264: 1, 139: 1, 656: 1, 1530: 1,

```
408: 1, 1654: 2, 1796: 1, 1146: 2, 1781: 1, 77: 1, 1440: 1, 1432: 1, 1633: 1,
1257: 1, 483: 1, 617: 1, 1493: 1, 624: 2, 1384: 1, 1244: 2, 1619: 1, 1655: 3,
1617: 1, 1170: 1, 1341: 1, 1568: 1, 1541: 1, 1477: 1, 1641: 2, 1029: 1, 1032:
1, 631: 1, 1873: 1, 1785: 1, 777: 1, 1844: 1, 1499: 2, 1525: 1, 809: 1, 973:
1, 635: 1, 980: 2, 1012: 2, 641: 1, 1761: 1, 625: 1, 76: 1, 1640: 1, 1733: 1,
46: 1, 1308: 1, 1103: 1, 1668: 1, 1710: 2, 204: 1, 213: 1, 914: 1, 191: 1, 71
0: 1, 1181: 1, 1284: 1, 1643: 2, 1053: 1, 190: 1, 1896: 1, 590: 1, 607: 1, 40
7: 1, 1799: 1, 706: 1, 53: 1, 1122: 1, 1835: 2, 1310: 1, 1742: 1, 908: 1, 57
5: 1, 1703: 1, 1931: 1, 1295: 1, 297: 1, 805: 2, 1231: 1, 1442: 1, 1984: 1, 6
40: 1, 557: 1, 248: 1, 1081: 1, 1481: 1, 1968: 2, 220: 1, 1323: 3, 1391: 1, 3
01: 1, 344: 2, 1104: 1, 500: 1, 1435: 1, 1644: 1, 1471: 1, 118: 1, 690: 2, 61
4: 1, 932: 1, 1570: 1, 1548: 1, 1678: 1, 243: 1, 1692: 1, 1877: 1, 764: 1, 94
4: 2, 1386: 1, 1200: 1, 851: 2, 1280: 1, 138: 1, 165: 1, 1993: 1, 1044: 1, 14
14: 1, 1567: 1, 1097: 1, 1513: 1, 1039: 1, 1254: 1, 1005: 1, 1515: 1, 726: 1,
212: 1, 1615: 1, 1780: 1, 1689: 1, 1381: 1, 1199: 1, 1445: 1, 1888: 1, 528:
1, 818: 2, 1881: 1, 903: 1, 1791: 1, 768: 1, 675: 1, 643: 1, 1863: 1, 1494:
1, 114: 1, 1748: 1, 1620: 1, 350: 1, 696: 1, 135: 1, 1255: 1, 1094: 1, 774:
1, 174: 1, 1694: 1, 1885: 1, 161: 1, 1359: 1, 1868: 1, 597: 1, 1639: 1, 1736:
1, 1634: 1, 1985: 1, 1163: 1, 436: 1, 499: 1, 315: 1, 251: 1, 832: 1, 586: 1,
402: 1, 1071: 1, 1631: 1, 1507: 1, 734: 1, 1479: 1, 38: 1, 716: 1, 1787: 1, 1
137: 1, 1398: 1, 1999: 1, 508: 1, 453: 1, 201: 1, 1982: 1}
<class 'dict'>
```

In [7]:

```
# Day 1 AM Lab 4 - attempt 1 - loops through all of the documents, and then fo
r each document, it counts the number of times that each pair of words co-occu
rs. Note that when you run this code, it will print out the total execution ti
me. Hint: when you call .get (index, default) to access an entry in a dictiona
ry, if that index in the dictionary is not occupied, the default value is retu
rned instead.

import time
start = time.time()
coOccurrences = {}
for doc in range (0, 50):
        for wordOne in wordsInCorpus[doc]:
                for wordTwo in wordsInCorpus[doc]:
                        coOccurrences [(wordOne, wordTwo)] = coOccurrences.get
((wordOne, wordTwo), 0) + 1
end = time.time()
end - start
```

Out[7]: 12.183886051177979

In [9]:
```
# Next, your task is to write a similar code, this time using the NumPy array-
based implementation. First, run the array-based code, which will compute the
 document corpus, and store it using NumPy arrays.
import numpy as np

# there are 2000 words in the corpus
alpha = np.full (2000, .1)

# there are 100 topics
beta = np.full (100, .1)

# this gets us the probabilty of each word happening in each of the 100 topics
wordsInTopic = np.random.dirichlet (alpha, 100)

# wordsInCorpus[i] will give us the vector of words in document i
wordsInCorpus = np.zeros ((50, 2000))

# generate each doc
for doc in range (0, 50):
        #
        # get the topic probabilities for this doc
        topicsInDoc = np.random.dirichlet (beta)
        #
        # assign each of the 1000 words in this doc to a topic
        wordsToTopic = np.random.multinomial (1000, topicsInDoc)
        #
        # and generate each of the 1000 words
        for topic in range (0, 100):
                wordsFromCurrentTopic = np.random.multinomial (wordsToTopic[to
pic], wordsInTopic[topic])
                wordsInCorpus[doc] = np.add (wordsInCorpus[doc], wordsFromCurr
entTopic)
```

In [10]:
```
# Day 1 AM Lab 4 - attempt 2 - using the NumPy array-based implementation. Fir
st, run the array-based code, which will compute the document corpus, and stor
e it using NumPy arrays.  We will now implement the co-occurrence analysis by
 looping through all of the documents, taking the outer product of each docume
nt with itself, and summing. It is important when you do this that you cap any
counts at one, since if wordA appears twice in a document, and wordB appears t
hree times, the number of documents where a co-occurrence occurred is still on
e, not six. You can cap the value in a NumPy array using np.clip (array, minTo
Allow, maxToAllow)).

import time
start = time.time()
coOccurrences = np.zeros ((2000, 2000))
for doc in range (0, 50):
        coOccurInThisDoc = np.outer (wordsInCorpus[doc], wordsInCorpus[doc])
        coOccurrences = np.add (coOccurrences, np.clip (coOccurInThisDoc, 0, 1
))
end = time.time()
end - start
```

Out[10]: 2.6083011627197266

In [11]:
```
# Day 1 AM Lab 4 - attempt 3 - one-line code that uses a matrix-multiply to co
mpute the answer.

import time
start = time.time()
res = np.dot (np.transpose (np.clip(wordsInCorpus, 0, 1)), np.clip (wordsInCor
pus, 0 , 1))
end = time.time()
end - start
```

Out[11]:
```
0.03308558464050293
```

In [5]:
```
# day1 PM lambdas
def addTwelveToResult (myLambda):
    return myLambda (3) + 12
a = 23
aCoolLambda = lambda x : x + a
addTwelveToResult (aCoolLambda) # prints 38
```

Out[5]:
```
38
```

In [6]:
```
a = 45
addTwelveToResult (aCoolLambda) # prints ???
```

Out[6]:
```
60
```

In [7]:
```
import numpy as np
def sumThem (myLambda):
    tot = 0
    for a in myLambda ():
        tot = tot + a
    return tot
x = np.array([1, 2, 3, 4, 5])
iter = lambda : (j for j in x)
sumThem (iter) # prints 15
```

Out[7]:
```
15
```

In [ ]:
```
def countWords (fileName):
    textFile = sc.textFile (fileName)
    lines = textFile.flatMap (lambda line: line.split(" "))
    counts = lines.map (lambda word: (word, 1))
    aggCounts = counts.reduceByKey (lambda a, b: a + b)
    retrun aggCounts.top (200, key=lambda p: p[1])
```