In [12]:

```
import pandas as pd
import numpy as np
import gensim
import nltk
from nltk.stem import WordNetLemmatizer,SnowballStemmer
from gensim.parsing.preprocessing import STOPWORDS
from nltk.stem.porter import *
import re
from gensim.utils import simple_preprocess
```

In [4]:

```
np.random.seed(2018)
```

In [5]:

```
data=pd.read_csv('C:/Users/nehal/Downloads/papers.csv',error_bad_lines=False)
```

In [6]:

```
data.head()
```

Out[6]:

	id	year	title	event_type	pdf_name	abstract	paper_text
0	1	1987	Self-Organization of Associative Database and ...	NaN	1-self-organization-of-associative-database-an...	Abstract Missing	767\n\nSELF-ORGANIZATION OF ASSOCIATIVE DATABA...
1	10	1987	A Mean Field Theory of Layer IV of Visual Cort...	NaN	10-a-mean-field-theory-of-layer-iv-of-visual-c...	Abstract Missing	683\n\nA MEAN FIELD THEORY OF LAYER IV OF VISU...
2	100	1988	Storing Covariance by the Associative Long-Ter...	NaN	100-storing-covariance-by-the-associative-long...	Abstract Missing	394\n\nSTORING COVARIANCE BY THE ASSOCIATIVE\n...
3	1000	1994	Bayesian Query Construction for Neural Network...	NaN	1000-bayesian-query-construction-for-neural-ne...	Abstract Missing	Bayesian Query Construction for Neural\nNetwor...
4	1001	1994	Neural Network Ensembles, Cross Validation, an...	NaN	1001-neural-network-ensembles-cross-validation...	Abstract Missing	Neural Network Ensembles, Cross\nValidation, a...

In [8]:

```
data=data[['id','paper_text']]
```

In [10]:

```
data.head()
```

Out[10]:

	id	paper_text
0	1	767\n\nSELF-ORGANIZATION OF ASSOCIATIVE DATABA...
1	10	683\n\nA MEAN FIELD THEORY OF LAYER IV OF VISU...
2	100	394\n\nSTORING COVARIANCE BY THE ASSOCIATIVE\n...
3	1000	Bayesian Query Construction for Neural\nNetwor...
4	1001	Neural Network Ensembles, Cross\nValidation, a...

In [18]:

```
def lem_stem(text):
```

```
    return SnowballStemmer('english').stem(WordNetLemmatizer().lemmatize(text))
```

In [19]:

```
def process(text):
    result=[]
    for token in simple_preprocess(text):
        if token not in STOPWORDS and len(token)>3:
            result.append(lem_stem(token))
    return result
```

In [20]:

```
clean_doc=data['paper_text'].map(process)
```

In [22]:

```
clean_doc[:10]
```

Out[22]:

```
0    [self, organ, associ, databas, applic, hisashi...
1    [mean, field, theori, layer, visual, cortex, a...
2    [store, covari, associ, long, term, potenti, d...
3    [bayesian, queri, construct, neural, network, ...
4    [neural, network, ensembl, cross, valid, activ...
5    [sing, neural, instanti, deform, model, christ...
6    [plastic, mediat, competit, learn, terrenc, se...
7    [iceg, morpholog, classif, analogu, vlsi, neur...
8    [real, time, control, tokamak, plasma, neural,...
9    [real, time, control, tokamak, plasma, neural,...
Name: paper_text, dtype: object
```

BoW

In [23]:

```
dictionary=gensim.corpora.Dictionary(clean_doc)
```

In [24]:

```
dictionary.filter_extremes(no_below=15,no_above=0.5,keep_n=100000)
```

In [26]:

```
bow_corpus=[dictionary.doc2bow(i) for i in clean_doc]
```

In [27]:

```
bow_corpus[4310]
```

Out[27]:

```
[(1, 1),
 (4, 1),
 (14, 1),
 (16, 3),
 (18, 1),
 (24, 2),
 (27, 1),
 (32, 2),
 (35, 1),
 (36, 1),
 (51, 1),
 (53, 1),
 (76, 1),
 (77, 3),
 (93, 2),
 (100, 1),
 (102, 2),
 (105, 1),
```

```
    (110, 1),
    (121, 1),
    (123, 3),
    (125, 4),
    (130, 1),
    (134, 3),
    (135, 2),
    (137, 6),
    (147, 5),
    (154, 1),
    (170, 2),
    (172, 1),
    (176, 1),
    (190, 1),
    (194, 1),
    (199, 1),
    (205, 18),
    (211, 1),
    (212, 1),
    (213, 2),
    (240, 2),
    (243, 2),
    (244, 4),
    (251, 1),
    (254, 1),
    (260, 1),
    (262, 4),
    (264, 1),
    (265, 7),
    (268, 1),
    (270, 1),
    (277, 7),
    (284, 1),
    (290, 2),
    (296, 2),
    (301, 4),
    (303, 4),
    (305, 1),
    (306, 1),
    (308, 4),
    (309, 1),
    (316, 37),
    (317, 2),
    (319, 5),
    (322, 18),
    (337, 5),
    (347, 1),
    (363, 1),
    (364, 1),
    (365, 1),
    (367, 2),
    (368, 3),
    (379, 1),
    (389, 1),
    (393, 1),
    (394, 3),
    (401, 6),
    (403, 4),
    (421, 1),
    (442, 4),
    (453, 1),
    (454, 3),
    (464, 1),
    (473, 1),
    (480, 1),
    (490, 1),
    (502, 2),
    (509, 1),
    (517, 1),
    (518, 1),
    (521, 3),
    (526, 1),
```

```
    (528, 2),
    (531, 1),
    (534, 1),
    (535, 4),
    (538, 1),
    (544, 3),
    (548, 2),
    (551, 2),
    (556, 1),
    (559, 40),
    (567, 1),
    (572, 3),
    (579, 1),
    (617, 2),
    (620, 6),
    (636, 2),
    (651, 2),
    (653, 1),
    (660, 3),
    (683, 3),
    (694, 5),
    (733, 4),
    (737, 2),
    (753, 1),
    (757, 1),
    (759, 1),
    (761, 1),
    (771, 1),
    (785, 2),
    (787, 7),
    (797, 1),
    (798, 1),
    (822, 21),
    (840, 7),
    (853, 2),
    (857, 3),
    (859, 1),
    (865, 1),
    (873, 27),
    (874, 1),
    (876, 1),
    (879, 5),
    (883, 4),
    (885, 1),
    (889, 1),
    (894, 2),
    (901, 4),
    (908, 1),
    (911, 2),
    (914, 1),
    (916, 2),
    (917, 1),
    (918, 1),
    (921, 2),
    (923, 1),
    (934, 1),
    (938, 1),
    (939, 3),
    (944, 1),
    (954, 1),
    (955, 1),
    (956, 2),
    (962, 7),
    (971, 1),
    (975, 1),
    (985, 1),
    (988, 1),
    (991, 4),
    (994, 1),
    (1002, 2),
    (1009, 2),
    (1013, 1),
```

```
    (1014, 3),
    (1019, 12),
    (1021, 2),
    (1023, 2),
    (1025, 13),
    (1026, 1),
    (1034, 2),
    (1040, 1),
    (1041, 9),
    (1044, 1),
    (1049, 1),
    (1053, 1),
    (1054, 1),
    (1110, 18),
    (1112, 1),
    (1129, 2),
    (1145, 2),
    (1156, 3),
    (1157, 2),
    (1165, 1),
    (1166, 1),
    (1174, 1),
    (1183, 1),
    (1193, 1),
    (1196, 2),
    (1208, 2),
    (1221, 1),
    (1224, 1),
    (1226, 4),
    (1228, 2),
    (1238, 1),
    (1242, 2),
    (1268, 1),
    (1274, 1),
    (1276, 1),
    (1283, 1),
    (1299, 31),
    (1312, 3),
    (1323, 8),
    (1334, 5),
    (1365, 2),
    (1386, 1),
    (1397, 2),
    (1412, 1),
    (1430, 3),
    (1441, 2),
    (1449, 4),
    (1453, 1),
    (1454, 1),
    (1458, 1),
    (1460, 2),
    (1480, 1),
    (1481, 1),
    (1508, 1),
    (1520, 2),
    (1557, 1),
    (1559, 1),
    (1563, 1),
    (1570, 2),
    (1595, 1),
    (1616, 1),
    (1633, 1),
    (1637, 1),
    (1647, 1),
    (1662, 3),
    (1676, 7),
    (1679, 1),
    (1701, 1),
    (1708, 3),
    (1737, 1),
    (1762, 1),
    (1765, 1),
```

```
 (1784, 2),
 (1789, 1),
 (1800, 1),
 (1811, 1),
 (1820, 2),
 (1823, 24),
 (1825, 2),
 (1844, 1),
 (1866, 1),
 (1873, 1),
 (1880, 1),
 (1882, 1),
 (1947, 1),
 (2009, 2),
 (2067, 1),
 (2068, 1),
 (2089, 1),
 (2111, 2),
 (2112, 1),
 (2119, 1),
 (2172, 2),
 (2201, 1),
 (2239, 2),
 (2247, 1),
 (2255, 2),
 (2276, 3),
 (2313, 1),
 (2329, 1),
 (2397, 1),
 (2403, 1),
 (2405, 1),
 (2425, 2),
 (2426, 1),
 (2479, 2),
 (2502, 5),
 (2519, 3),
 (2579, 3),
 (2581, 1),
 (2589, 1),
 (2619, 1),
 (2658, 1),
 (2674, 1),
 (2675, 5),
 (2690, 1),
 (2698, 1),
 (2700, 5),
 (2726, 2),
 (2741, 1),
 (2780, 2),
 (2804, 1),
 (2810, 2),
 (2821, 2),
 (2825, 1),
 (2839, 21),
 (2843, 1),
 (2854, 4),
 (2856, 2),
 (2871, 1),
 (2885, 2),
 (2970, 2),
 (2982, 1),
 (3025, 3),
 (3053, 1),
 (3056, 3),
 (3065, 1),
 (3082, 1),
 (3085, 1),
 (3134, 1),
 (3141, 6),
 (3146, 1),
 (3147, 1),
 (3180, 1),
```

```
 (3194, 1),
 (3332, 1),
 (3383, 2),
 (3460, 1),
 (3472, 1),
 (3488, 1),
 (3498, 1),
 (3548, 1),
 (3564, 1),
 (3567, 1),
 (3610, 1),
 (3616, 9),
 (3658, 1),
 (3675, 1),
 (3694, 8),
 (3864, 1),
 (3868, 2),
 (3873, 1),
 (3944, 1),
 (3989, 1),
 (4095, 1),
 (4171, 1),
 (4329, 1),
 (4400, 1),
 (4455, 1),
 (4470, 1),
 (4524, 1),
 (4635, 9),
 (4700, 1),
 (4738, 1),
 (4847, 2),
 (5103, 1),
 (5304, 1),
 (5307, 1),
 (5312, 1),
 (5313, 1),
 (5318, 1),
 (5674, 3),
 (6071, 2),
 (6157, 1),
 (6160, 1),
 (6277, 1),
 (6333, 1),
 (6387, 1),
 (6446, 1),
 (6468, 1),
 (6553, 3),
 (6560, 2),
 (6577, 1),
 (6650, 1),
 (6851, 5),
 (6940, 3),
 (7119, 1),
 (7171, 1),
 (7462, 1),
 (7471, 1),
 (7595, 3),
 (7604, 1),
 (7801, 1),
 (7841, 1),
 (7846, 2),
 (8106, 1),
 (8251, 1),
 (8467, 8),
 (8651, 1),
 (8709, 2),
 (8908, 1),
 (8936, 1),
 (9015, 2),
 (9016, 1),
 (9085, 1),
 (9218, 1),
```

```
 (9292, 1),
 (9293, 1),
 (9501, 1),
 (9663, 1),
 (9747, 2),
 (9798, 2),
 (10013, 1),
 (10046, 1),
 (10058, 1),
 (10346, 1),
 (10380, 1),
 (10396, 1),
 (10475, 1),
 (10572, 1),
 (10625, 1),
 (10731, 1),
 (10974, 12)]
```

In [28]:

```python
from gensim import corpora,models
```

In [29]:

```python
tfidf=models.TfidfModel(bow_corpus)
```

In [31]:

```python
corpus_tfidf=tfidf[bow_corpus]
```

In [32]:

```python
from pprint import pprint
for doc in corpus_tfidf:
    pprint(doc)
    break
```

```
[(0, 0.04641498364526951),
 (1, 0.016180935748663635),
 (2, 0.030941689773980735),
 (3, 0.018410338822421806),
 (4, 0.02044348103201744),
 (5, 0.01375843310177131),
 (6, 0.016580197020705605),
 (7, 0.012630296856504426),
 (8, 0.013345574293887243),
 (9, 0.02458912052166688),
 (10, 0.010078657302783572),
 (11, 0.016329357413836454),
 (12, 0.02659666249110095),
 (13, 0.028467924504446716),
 (14, 0.006837731608066944),
 (15, 0.007348096823887533),
 (16, 0.014472960566842557),
 (17, 0.019345666070698684),
 (18, 0.06133744909981713),
 (19, 0.04685736211024454),
 (20, 0.010161644995211763),
 (21, 0.018561756307282903),
 (22, 0.016525757058043263),
 (23, 0.01401533064242998),
 (24, 0.010756014939245386),
 (25, 0.0075779725849975),
 (26, 0.027740048647153844),
 (27, 0.049795287314366325),
 (28, 0.016648336398781927),
 (29, 0.07564241895346906),
 (30, 0.009846338226125758),
 (31, 0.1137507489750968),
 (32, 0.03199582552906563),
 (33, 0.013770692378516088),
 (34, 0.011631751383955836),
```

```
(34, 0.011631751383955826),
(35, 0.021430980953547542),
(36, 0.11451399345719121),
(37, 0.0774171766905939),
(38, 0.023425413023210877),
(39, 0.020796121772866077),
(40, 0.03226616439235617),
(41, 0.024470361835282586),
(42, 0.0073040282856693985),
(43, 0.023165236722228148),
(44, 0.25342192224026217),
(45, 0.05603460883126851),
(46, 0.016830029752567643),
(47, 0.029751392478429284),
(48, 0.031667964233606925),
(49, 0.013582982706071686),
(50, 0.007154428342005901),
(51, 0.020875016606033223),
(52, 0.059059492785417),
(53, 0.014996536416776922),
(54, 0.027255004604922188),
(55, 0.017631897469754645),
(56, 0.02605162669387783),
(57, 0.008440981388190195),
(58, 0.031232696027390292),
(59, 0.03332516353341918),
(60, 0.02071763030777255),
(61, 0.007140970651057846),
(62, 0.06295683117920021),
(63, 0.017880532489673615),
(64, 0.06853807644213185),
(65, 0.029113263652480285),
(66, 0.06448255485073938),
(67, 0.047326567347708674),
(68, 0.022977903665488606),
(69, 0.052989758230061114),
(70, 0.01418035499988457),
(71, 0.008933483145266622),
(72, 0.01930841206420031),
(73, 0.024462032015582027),
(74, 0.14347819281611401),
(75, 0.02173764995599897),
(76, 0.007666686811838701),
(77, 0.02558198721601355),
(78, 0.04833054600031651),
(79, 0.013638819136652528),
(80, 0.011960515600202058),
(81, 0.023851012841532595),
(82, 0.029209506871772293),
(83, 0.00847924854211587),
(84, 0.038056924198546616),
(85, 0.031293398425736245),
(86, 0.24070438912306832),
(87, 0.013150226231999285),
(88, 0.013027244154942153),
(89, 0.07611384839709323),
(90, 0.03431056976397554),
(91, 0.03819481648428513),
(92, 0.04804210848249543),
(93, 0.01467411228564477),
(94, 0.009255360516611381),
(95, 0.006750307572896785),
(96, 0.017573344419582367),
(97, 0.03314327357848812),
(98, 0.007106089777496567),
(99, 0.009199021660996263),
(100, 0.014410795444559752),
(101, 0.024068829183777507),
(102, 0.013322664055362322),
(103, 0.024563535321129867),
(104, 0.012885896353091262),
(105, 0.010611505772420735),
(106, 0.044636132261045857 5),
```

```
(106, 0.044636132610458575),
(107, 0.03093612655964698),
(108, 0.031354580689917005),
(109, 0.02344714626589315),
(110, 0.008657101944435908),
(111, 0.006648704326863557),
(112, 0.04210293835157814),
(113, 0.05798214898448388),
(114, 0.017580180441519728),
(115, 0.02418957291372838),
(116, 0.009209554004804733),
(117, 0.047326567347708674),
(118, 0.06981774789982173),
(119, 0.025919375356223642),
(120, 0.010375134645677046),
(121, 0.008289777241063702),
(122, 0.059330467865016584),
(123, 0.02825423300396953),
(124, 0.018910604738367264),
(125, 0.06567114249444816),
(126, 0.0219084849709991),
(127, 0.020702497452368674),
(128, 0.016444808446752723),
(129, 0.0054711695839146195),
(130, 0.012478749041653529),
(131, 0.018692233998648517),
(132, 0.02762538705198685),
(133, 0.014417678073838746),
(134, 0.006991821884537586),
(135, 0.009576458505110628),
(136, 0.019907773165239045),
(137, 0.03676811538128344),
(138, 0.016224294495758292),
(139, 0.03612811456336562),
(140, 0.01585911519424252),
(141, 0.01845427964989153),
(142, 0.033106729163341855),
(143, 0.0230000604355161),
(144, 0.012923440579979597),
(145, 0.0327656613166855),
(146, 0.039786529104962796),
(147, 0.022931085863165373),
(148, 0.04063190574156002),
(149, 0.019511661820822816),
(150, 0.01055277745507229),
(151, 0.006732430444737371),
(152, 0.025919375356223642),
(153, 0.040442881330888074),
(154, 0.008789066200536598),
(155, 0.06435305035118268),
(156, 0.04094654978038202),
(157, 0.04194152883658442),
(158, 0.02511913232847311),
(159, 0.028637206406713373),
(160, 0.04522193197473524),
(161, 0.02620798587271176),
(162, 0.01606080780975109),
(163, 0.026303340460167458),
(164, 0.00956541519843673),
(165, 0.011679942798153606),
(166, 0.13098846466306807),
(167, 0.02331300704512283),
(168, 0.14681578813878157),
(169, 0.027291255332152084),
(170, 0.00664365900979168),
(171, 0.02421452932616025),
(172, 0.012267273696315188),
(173, 0.018410338822421806),
(174, 0.03645591184725241),
(175, 0.031293398425736245),
(176, 0.03442444717896831),
(177, 0.01672727063620114),
(178, 0.00801390333012004),
```

```
(178, 0.008013993333012004),
(179, 0.02322946544034085),
(180, 0.01374539981968559),
(181, 0.01927592229366988),
(182, 0.02686706610260949),
(183, 0.009405231348455678),
(184, 0.021806547490667218),
(185, 0.00780023016299763),
(186, 0.02487628559426228),
(187, 0.04559954423653786),
(188, 0.025202208113721477),
(189, 0.025173387666979304),
(190, 0.011453506521260193),
(191, 0.03722689568820158),
(192, 0.006866228603360115),
(193, 0.014733907467329504),
(194, 0.013144382882447138),
(195, 0.02859408921378031),
(196, 0.08683783797160823),
(197, 0.017305114741217993),
(198, 0.008110927874367018),
(199, 0.19456984870104266),
(200, 0.007367455995245939),
(201, 0.008524134014070362),
(202, 0.02105146917578907),
(203, 0.005702151397043387),
(204, 0.015199527105417865),
(205, 0.006994460206061063),
(206, 0.0688021829043896),
(207, 0.020938716686727997),
(208, 0.019884583599511056),
(209, 0.03421533479489052),
(210, 0.02477207574527468),
(211, 0.013262096423280769),
(212, 0.014062982908576186),
(213, 0.006889622624539601),
(214, 0.031541084078675334),
(215, 0.03082109357388534),
(216, 0.03224127742536969),
(217, 0.13348479999943777),
(218, 0.01438332597077504),
(219, 0.08030403904875545),
(220, 0.059602973217489495),
(221, 0.023564264344633955),
(222, 0.07415307048491714),
(223, 0.06974059337931787),
(224, 0.04959381049188654),
(225, 0.0076551825122635085),
(226, 0.007803159024381373),
(227, 0.012179117450778607),
(228, 0.006042229073374155),
(229, 0.03886456294500761),
(230, 0.0699758501524303),
(231, 0.02495648855151167),
(232, 0.014668596863449108),
(233, 0.01115661385900725),
(234, 0.18904126658287657),
(235, 0.009333502233420209),
(236, 0.06471463683207924),
(237, 0.10465512187802102),
(238, 0.014591763598385226),
(239, 0.0890357088627535),
(240, 0.005408571008851649),
(241, 0.023949940269735723),
(242, 0.010000440203367676),
(243, 0.008113976728499091),
(244, 0.014791504132134491),
(245, 0.03684303936303956),
(246, 0.04054584222525922),
(247, 0.017456577816068183),
(248, 0.027655809335631158),
(249, 0.01143256166624225),
(250, 0.01776558702040225),
```

```
    (250, 0.01776559702040225),
    (251, 0.011284166158113139),
    (252, 0.04559954423653786),
    (253, 0.0334592160628904 4),
    (254, 0.021422911953173536),
    (255, 0.0211330243678832),
    (256, 0.03518057413567203),
    (257, 0.013969563659505949),
    (258, 0.007989948215064187),
    (259, 0.025513528611653694),
    (260, 0.016329136575476595),
    (261, 0.006506186192722136),
    (262, 0.013608376123632434),
    (263, 0.0108901670256658),
    (264, 0.05335832654759435),
    (265, 0.009488550136095323),
    (266, 0.014254161087822678),
    (267, 0.03826166079374692),
    (268, 0.012468036906913438),
    (269, 0.017779321281676305),
    (270, 0.009751559719686862),
    (271, 0.06340329734106522),
    (272, 0.03402160220054985),
    (273, 0.05078836478544026),
    (274, 0.10629014769218381),
    (275, 0.08168408335530147),
    (276, 0.025173387666979304),
    (277, 0.07090177499942285),
    (278, 0.02096411692103243),
    (279, 0.11375298774246272),
    (280, 0.019754807977817174),
    (281, 0.015402605340556562),
    (282, 0.015976129545860192),
    (283, 0.0745492047378013),
    (284, 0.008601691400766259),
    (285, 0.010586281906158585),
    (286, 0.009853970885634089),
    (287, 0.036866883745507534),
    (288, 0.024666387547257435),
    (289, 0.4720322147835124),
    (290, 0.01982291406682984),
    (291, 0.04246106992192818),
    (292, 0.011098100227873702),
    (293, 0.14384143439021488),
    (294, 0.04294273614329879),
    (295, 0.02262984401336354),
    (296, 0.0073012823157579685),
    (297, 0.06481123781209504),
    (298, 0.088591102540719),
    (299, 0.035473681296149785),
    (300, 0.028888280240200768),
    (301, 0.14375699909245737),
    (302, 0.015682670857328898),
    (303, 0.022365199777677036),
    (304, 0.012581260687521231),
    (305, 0.01591740122851175),
    (306, 0.010306013212275866),
    (307, 0.0789945315813156),
    (308, 0.008178281683355612),
    (309, 0.018561756307282903),
    (310, 0.04246144049859854),
    (311, 0.0400438115900541),
    (312, 0.01013784328671134),
    (313, 0.01964579890653848),
    (314, 0.030389776867430694),
    (315, 0.034492684427856315),
    (316, 0.005600109866957533),
    (317, 0.03442444717896831),
    (318, 0.017843372815092993),
    (319, 0.014974307058898262),
    (320, 0.059011345832438944),
    (321, 0.02704282210373398),
    (322, 0.0066512282193086116),
```

```
 (322, 0.006651228219308611),
 (323, 0.058654978227425),
 (324, 0.029361900680734593),
 (325, 0.04629447674667579),
 (326, 0.011040025664197593),
 (327, 0.00635166053512833),
 (328, 0.015260709369598622),
 (329, 0.021606330011331547),
 (330, 0.016820631431905552),
 (331, 0.005914950562459972),
 (332, 0.01118377008376023),
 (333, 0.03721789724985936),
 (334, 0.02793500898388258),
 (335, 0.013632595158742374),
 (336, 0.036102241723753016),
 (337, 0.019022144931550362),
 (338, 0.022110476690149893),
 (339, 0.016580197020705605),
 (340, 0.034132642017218336),
 (341, 0.01700118081797451),
 (342, 0.03059462192101937),
 (343, 0.02247229117088472),
 (344, 0.03387319260199469),
 (345, 0.11707006067243282),
 (346, 0.02565928651875511),
 (347, 0.033704935625100255),
 (348, 0.025119431751256634),
 (349, 0.0432768799073302),
 (350, 0.016094940240438057),
 (351, 0.03431056976397554),
 (352, 0.019097408242142566),
 (353, 0.01493003897187619),
 (354, 0.014655405478258712),
 (355, 0.01591740122851175),
 (356, 0.012382869432753563),
 (357, 0.054494511860472265),
 (358, 0.037973022060680546),
 (359, 0.029314564043175183),
 (360, 0.018038770696899954),
 (361, 0.030994538279279026),
 (362, 0.014751535259204272),
 (363, 0.008327302241188873),
 (364, 0.009322799973893903),
 (365, 0.011138637825126255),
 (366, 0.016610781106049008),
 (367, 0.007359153277359562),
 (368, 0.010873139690843688),
 (369, 0.013171746068554443),
 (370, 0.05684796622211675),
 (371, 0.031952259091720385),
 (372, 0.030109057522710318),
 (373, 0.017528287942317027),
 (374, 0.019405459606097047),
 (375, 0.016610781106049008),
 (376, 0.024512617268301867),
 (377, 0.012537932282605477),
 (378, 0.044790116915628564),
 (379, 0.007005022495130176),
 (380, 0.038056924198546616),
 (381, 0.03038382643564956),
 (382, 0.1687166104682213),
 (383, 0.038665852294940 24),
 (384, 0.020094551084286455),
 (385, 0.030484934985635287)]
```

In [ ]:

```

```

In [33]:

```
lda_model=gensim.models.LdaMulticore(bow_corpus,num_topics=10,id2word=dictionary,passes=
```

```
2,workers=2)
```

In [34]:

```
for id,topic in lda_model.print_topics(-1):
    print(id,'\n', topic)
```

```
0
 0.007*"unit" + 0.005*"loss" + 0.005*"theorem" + 0.005*"decis" + 0.004*"sequenc" + 0.004*
"kernel" + 0.004*"polici" + 0.004*"gradient" + 0.004*"classifi" + 0.004*"nois"
1
 0.016*"imag" + 0.009*"word" + 0.007*"layer" + 0.007*"infer" + 0.006*"label" + 0.006*"top
ic" + 0.005*"deep" + 0.005*"dataset" + 0.005*"gradient" + 0.004*"prior"
2
 0.006*"loss" + 0.006*"gradient" + 0.006*"theorem" + 0.006*"convex" + 0.005*"regret" + 0.
005*"label" + 0.004*"graph" + 0.004*"cost" + 0.004*"unit" + 0.004*"regular"
3
 0.010*"neuron" + 0.007*"spike" + 0.006*"kernel" + 0.006*"activ" + 0.006*"densiti" + 0.00
5*"signal" + 0.004*"rule" + 0.004*"prior" + 0.004*"theorem" + 0.003*"dynam"
4
 0.014*"cluster" + 0.012*"label" + 0.011*"imag" + 0.009*"graph" + 0.007*"dataset" + 0.005
*"loss" + 0.004*"infer" + 0.004*"nois" + 0.004*"spars" + 0.004*"classif"
5
 0.008*"action" + 0.007*"agent" + 0.007*"game" + 0.006*"regret" + 0.005*"convex" + 0.005*
"layer" + 0.005*"gradient" + 0.005*"reward" + 0.005*"player" + 0.004*"imag"
6
 0.016*"rank" + 0.008*"norm" + 0.007*"polici" + 0.006*"spars" + 0.005*"action" + 0.005*"t
heorem" + 0.005*"dynam" + 0.004*"convex" + 0.004*"group" + 0.004*"tensor"
7
 0.012*"node" + 0.009*"kernel" + 0.008*"imag" + 0.008*"graph" + 0.006*"cluster" + 0.006*"
tree" + 0.005*"layer" + 0.004*"dynam" + 0.004*"dataset" + 0.004*"polici"
8
 0.007*"regular" + 0.007*"neuron" + 0.006*"constraint" + 0.006*"theorem" + 0.006*"loss" +
0.005*"convex" + 0.005*"gradient" + 0.005*"stochast" + 0.005*"activ" + 0.004*"regress"
9
 0.011*"imag" + 0.010*"kernel" + 0.005*"signal" + 0.005*"visual" + 0.005*"neuron" + 0.004
*"layer" + 0.004*"dataset" + 0.004*"infer" + 0.004*"rank" + 0.004*"respons"
```

In [35]:

```
lda_model_tfidf=gensim.models.LdaMulticore(corpus_tfidf,num_topics=10,id2word=dictionary
,passes=2,workers=2)
```

In [36]:

```
for id, topic in lda_model_tfidf.print_topics(-1):
    print(id,'\n', topic)
```

```
0
 0.009*"polici" + 0.009*"regret" + 0.007*"reward" + 0.007*"action" + 0.006*"bandit" + 0.0
06*"agent" + 0.004*"arm" + 0.003*"game" + 0.003*"player" + 0.002*"pomdp"
1
 0.002*"hash" + 0.002*"kernel" + 0.002*"imag" + 0.001*"graph" + 0.001*"cluster" + 0.001*"
node" + 0.001*"tree" + 0.001*"regress" + 0.001*"lasso" + 0.001*"convex"
2
 0.005*"neuron" + 0.003*"imag" + 0.003*"layer" + 0.003*"spike" + 0.002*"cell" + 0.002*"po
lici" + 0.002*"stimulus" + 0.002*"posterior" + 0.002*"activ" + 0.002*"unit"
3
 0.002*"spike" + 0.002*"tensor" + 0.002*"graph" + 0.002*"rank" + 0.002*"neuron" + 0.002*"
kernel" + 0.002*"imag" + 0.002*"lift" + 0.002*"label" + 0.002*"infer"
4
 0.005*"cluster" + 0.004*"imag" + 0.002*"layer" + 0.002*"label" + 0.002*"convolut" + 0.00
2*"rank" + 0.002*"deep" + 0.002*"video" + 0.002*"dataset" + 0.002*"loss"
5
 0.004*"convex" + 0.004*"kernel" + 0.003*"theorem" + 0.003*"cluster" + 0.003*"graph" + 0.
003*"rank" + 0.003*"loss" + 0.003*"node" + 0.003*"norm" + 0.002*"lemma"
6
 0.004*"manifold" + 0.003*"kernel" + 0.003*"cluster" + 0.002*"graph" + 0.002*"causal" + 0
.002*"worker" + 0.002*"rank" + 0.002*"label" + 0.002*"imag" + 0.002*"theorem"
7
 0.003*"kernel" + 0.001*"tensor" + 0.001*"imag" + 0.001*"topic" + 0.001*"neuron" + 0.001*
"node" + 0.001*"stimulus" + 0.001*"cluster" + 0.001*"regress" + 0.001*"theorem"
```

```
8
 0.002*"kernel" + 0.002*"polici" + 0.002*"tensor" + 0.002*"imag" + 0.001*"layer" + 0.001*
"neuron" + 0.001*"cluster" + 0.001*"rank" + 0.001*"cell" + 0.001*"latent"
9
 0.005*"privaci" + 0.004*"privat" + 0.003*"rank" + 0.002*"label" + 0.002*"cluster" + 0.00
2*"queri" + 0.002*"user" + 0.001*"graph" + 0.001*"activ" + 0.001*"posterior"
```

In [39]:

```
for id, score in sorted(lda_model[bow_corpus[4310]],key=lambda tup:-1*tup[1]):
    print(score,'\n',lda_model.print_topic(id,5))
```

```
0.6379622
 0.006*"loss" + 0.006*"gradient" + 0.006*"theorem" + 0.006*"convex" + 0.005*"regret"
0.36035556
 0.007*"regular" + 0.007*"neuron" + 0.006*"constraint" + 0.006*"theorem" + 0.006*"loss"
```

In [42]:

```
for id , score in sorted(lda_model_tfidf[bow_corpus[4310]],key=lambda tup: -1*tup[1]):
    print(score, '\n', lda_model_tfidf.print_topic(id,5))
```

```
0.99554706
 0.004*"convex" + 0.004*"kernel" + 0.003*"theorem" + 0.003*"cluster" + 0.003*"graph"
```

In [43]:

```
unseen='How a Pentagon deal became an identity crisis for Google'
bow=dictionary.doc2bow(process(unseen))

for id, score in sorted(lda_model_tfidf[bow],key=lambda tup: -1*tup[1]):
    print(score,'\n',lda_model_tfidf.print_topic(id,5))
```

```
0.5755596
 0.004*"convex" + 0.004*"kernel" + 0.003*"theorem" + 0.003*"cluster" + 0.003*"graph"
0.2641328
 0.003*"kernel" + 0.001*"tensor" + 0.001*"imag" + 0.001*"topic" + 0.001*"neuron"
0.020041779
 0.005*"neuron" + 0.003*"imag" + 0.003*"layer" + 0.003*"spike" + 0.002*"cell"
0.020039253
 0.005*"cluster" + 0.004*"imag" + 0.002*"layer" + 0.002*"label" + 0.002*"convolut"
0.020038994
 0.009*"polici" + 0.009*"regret" + 0.007*"reward" + 0.007*"action" + 0.006*"bandit"
0.02003833
 0.004*"manifold" + 0.003*"kernel" + 0.003*"cluster" + 0.002*"graph" + 0.002*"causal"
0.020037597
 0.002*"spike" + 0.002*"tensor" + 0.002*"graph" + 0.002*"rank" + 0.002*"neuron"
0.020037483
 0.005*"privaci" + 0.004*"privat" + 0.003*"rank" + 0.002*"label" + 0.002*"cluster"
0.020037124
 0.002*"hash" + 0.002*"kernel" + 0.002*"imag" + 0.001*"graph" + 0.001*"cluster"
0.020036994
 0.002*"kernel" + 0.002*"polici" + 0.002*"tensor" + 0.002*"imag" + 0.001*"layer"
```

In [ ]:

```

```

In [45]:

```
lsa_model=gensim.models.LsiModel(bow_corpus,num_topics=10,id2word=dictionary)
```

In [54]:

```
for id,topic in lsa_model.print_topics(-1):
    print(id,'\n',topic)
```

```
0
 0.322*"imag" + 0.175*"label" + 0.158*"cluster" + 0.155*"kernel" + 0.136*"dataset" + 0.13
1*"graph" + 0.126*"layer" + 0.119*"infer" + 0.116*"loss" + 0.112*"node"
1
```

```
 0.709*"imag" + -0.201*"polici" + -0.151*"graph" + -0.144*"theorem" + -0.144*"cluster" +
0.138*"layer" + -0.129*"action" + -0.119*"node" + 0.107*"visual" + -0.104*"reward"
2
 -0.448*"cluster" + 0.322*"neuron" + 0.272*"polici" + -0.232*"graph" + -0.216*"label" + 0
.213*"action" + -0.194*"kernel" + 0.190*"spike" + 0.157*"agent" + 0.155*"reward"
3
 0.427*"neuron" + -0.331*"polici" + 0.328*"cluster" + 0.285*"spike" + -0.246*"imag" + -0.
225*"action" + -0.166*"agent" + -0.153*"reward" + 0.140*"stimulus" + 0.139*"activ"
4
 0.559*"cluster" + -0.514*"kernel" + 0.218*"polici" + -0.182*"loss" + 0.171*"action" + 0.
145*"agent" + 0.142*"imag" + 0.133*"node" + -0.127*"convex" + 0.122*"graph"
5
 0.603*"kernel" + 0.375*"cluster" + -0.367*"label" + -0.226*"node" + 0.214*"polici" + -0.
162*"tree" + -0.156*"graph" + 0.142*"imag" + 0.130*"action" + -0.118*"word"
6
 -0.427*"graph" + -0.415*"kernel" + -0.360*"node" + 0.308*"cluster" + 0.242*"loss" + -0.2
32*"tree" + 0.195*"convex" + 0.172*"rank" + 0.150*"gradient" + -0.148*"edg"
7
 -0.648*"label" + -0.190*"classifi" + -0.164*"neuron" + 0.160*"gradient" + 0.150*"graph"
+ 0.150*"infer" + 0.147*"node" + -0.142*"cluster" + -0.141*"classif" + -0.136*"kernel"
8
 -0.358*"word" + -0.321*"topic" + 0.301*"graph" + -0.219*"infer" + 0.210*"imag" + 0.185*"
loss" + -0.181*"document" + 0.160*"convex" + -0.156*"latent" + -0.152*"posterior"
9
 0.639*"layer" + 0.269*"unit" + -0.225*"imag" + 0.203*"deep" + -0.164*"infer" + 0.159*"hi
dden" + -0.149*"prior" + 0.147*"convolut" + 0.134*"architectur" + -0.132*"spike"
```

In [55]:

```
nmf_model=gensim.models.Nmf(bow_corpus,num_topics=10,id2word=dictionary)
```

In [56]:

```
for id,topic in nmf_model.print_topics(-1):
    print(id,'\n',topic)
```

```
0
 0.031*"activ" + 0.023*"cell" + 0.019*"unit" + 0.013*"dynam" + 0.012*"neuron" + 0.012*"me
mori" + 0.011*"field" + 0.008*"simul" + 0.007*"code" + 0.007*"layer"
1
 0.019*"stimulus" + 0.016*"respons" + 0.015*"nois" + 0.014*"reward" + 0.014*"region" + 0.
013*"decis" + 0.011*"subject" + 0.011*"polici" + 0.011*"human" + 0.011*"trial"
2
 0.064*"neuron" + 0.053*"spike" + 0.012*"signal" + 0.012*"fire" + 0.012*"synapt" + 0.011*
"stimulus" + 0.010*"respons" + 0.010*"correl" + 0.009*"cell" + 0.009*"popul"
3
 0.068*"cluster" + 0.027*"label" + 0.018*"graph" + 0.009*"classifi" + 0.009*"distanc" + 0
.009*"dataset" + 0.009*"queri" + 0.008*"topic" + 0.008*"classif" + 0.007*"partit"
4
 0.066*"kernel" + 0.014*"graph" + 0.013*"tree" + 0.012*"node" + 0.007*"regress" + 0.007*"
word" + 0.006*"dataset" + 0.006*"covari" + 0.006*"edg" + 0.006*"regular"
5
 0.011*"loss" + 0.010*"theorem" + 0.010*"convex" + 0.008*"rank" + 0.008*"gradient" + 0.00
7*"norm" + 0.006*"infer" + 0.006*"stochast" + 0.006*"prior" + 0.006*"bayesian"
6
 0.028*"label" + 0.024*"node" + 0.022*"tree" + 0.014*"agent" + 0.009*"infer" + 0.008*"cos
t" + 0.007*"game" + 0.006*"action" + 0.006*"decis" + 0.006*"dynam"
7
 0.025*"layer" + 0.013*"unit" + 0.011*"sequenc" + 0.011*"hidden" + 0.010*"deep" + 0.008*"
word" + 0.008*"classif" + 0.007*"recognit" + 0.007*"gradient" + 0.007*"classifi"
8
 0.080*"imag" + 0.013*"visual" + 0.009*"pixel" + 0.009*"segment" + 0.008*"detect" + 0.008
*"scene" + 0.007*"dataset" + 0.007*"patch" + 0.007*"spatial" + 0.007*"filter"
9
 0.052*"polici" + 0.037*"action" + 0.019*"graph" + 0.018*"reward" + 0.013*"agent" + 0.011
*"gradient" + 0.010*"regret" + 0.009*"reinforc" + 0.008*"game" + 0.007*"plan"
```

In [ ]: