

```
In [1]: import pandas as pd
import re
import nltk
import gensim
import matplotlib.pyplot as plt
from nltk.tokenize import word_tokenize
from gensim.models import Word2Vec
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

nltk.download('punkt')

df = pd.read_csv("email_dataset.csv", quotechar="'")

def clean_text(text):
    if isinstance(text, float):
        return ""
    text = text.lower()
    text = re.sub(r'\W', ' ', text)
    text = re.sub(r'\s+', ' ', text).strip()
    return text
```

```
[nltk_data] Downloading package punkt to /home/aditya/nltk_data...
[nltk_data] Package punkt is already up-to-date!
```

```
In [2]: df
```

```
Out[2]:
```

	category	message
0	spam	Congratulations! You've won a \$1000 gift card....
1	ham	Reminder: Your appointment is scheduled for to...
2	spam	URGENT: Your account has been compromised. Res...
3	ham	Join us for a webinar on cybersecurity best pr...
4	spam	Limited-time offer! Buy one, get one free on a...
...
67	ham	Please find the attached report for your review.
68	spam	Warning! Unusual activity detected on your ban...
69	ham	Traffic is heavy on the way to office. Leave e...
70	spam	Congratulations! You've unlocked a secret rewa...
71	ham	How was your weekend? Let's catch up soon.

72 rows × 2 columns

```
In [3]: df['cleaned_message'] = df['message'].astype(str).apply(clean_text)

df['tokens'] = df['cleaned_message'].apply(word_tokenize)

corpus = df['tokens'].tolist()
```

```
In [23]: df
```

Out[23]:

	category	message	cleaned_message	tokens
0	spam	Congratulations! You've won a \$1000 gift card....	congratulations you ve won a 1000 gift card cl...	[congratulations, you, ve, won, a, 1000, gift,...
1	ham	Reminder: Your appointment is scheduled for to...	reminder your appointment is scheduled for tom...	[reminder, your, appointment, is, scheduled, f...
2	spam	URGENT: Your account has been compromised. Res...	urgent your account has been compromised reset...	[urgent, your, account, has, been, compromised...
3	ham	Join us for a webinar on cybersecurity best pr...	join us for a webinar on cybersecurity best pr...	[join, us, for, a, webinar, on, cybersecurity,...
4	spam	Limited-time offer! Buy one, get one free on a...	limited time offer buy one get one free on all...	[limited, time, offer, buy, one, get, one, fre...
...
67	ham	Please find the attached report for your review.	please find the attached report for your review	[please, find, the, attached, report, for, you...
68	spam	Warning! Unusual activity detected on your ban...	warning unusual activity detected on your bank...	[warning, unusual, activity, detected, on, you...
69	ham	Traffic is heavy on the way to office. Leave e...	traffic is heavy on the way to office leave early	[traffic, is, heavy, on, the, way, to, office,...
70	spam	Congratulations! You've unlocked a secret rewa...	congratulations you ve unlocked a secret rewar...	[congratulations, you, ve, unlocked, a, secret...
71	ham	How was your weekend? Let's catch up soon.	how was your weekend let s catch up soon	[how, was, your, weekend, let, s, catch, up, s...

72 rows × 4 columns

```
In [4]: corpus
```

```
Out[4]: [['congratulations',  
         'you',  
         've',  
         'won',  
         'a',  
         '1000',  
         'gift',  
         'card',  
         'click',  
         'here',  
         'to',  
         'claim'],  
 ['reminder',  
  'your',  
  'appointment',  
  'is',  
  'scheduled',  
  'for',  
  'tomorrow',  
  'at',  
  '10',  
  'am'],  
 ['urgent',  
  'your',  
  'account',  
  'has',  
  'been',  
  'compromised',  
  'reset',  
  'your',  
  'password',  
  'now'],  
 ['join',  
  'us',  
  'for',  
  'a',  
  'webinar',  
  'on',  
  'cybersecurity',  
  'best',  
  'practices'],  
 ['limited',  
  'time',  
  'offer',  
  'buy',  
  'one',  
  'get',  
  'one',  
  'free',  
  'on',  
  'all',  
  'products'],  
 ['hello', 'just', 'checking', 'in', 'to', 'see', 'how', 'you', 're', 'doing'],  
 ['don',  
  't',  
  'miss',
```

```
'out',  
'on',  
'our',  
'biggest',  
'sale',  
'of',  
'the',  
'year'],  
['can', 'we', 'schedule', 'a', 'meeting', 'for', 'next', 'week'],  
['win', 'a', 'brand', 'new', 'iphone', 'enter', 'the', 'contest', 'now'],  
['your', 'package', 'has', 'been', 'shipped', 'track', 'it', 'here'],  
['you',  
'are',  
'the',  
'lucky',  
'winner',  
'of',  
'a',  
'free',  
'vacation',  
'claim',  
'your',  
'reward',  
'now'],  
['meeting',  
'is',  
'scheduled',  
'for',  
'3',  
'pm',  
'please',  
'be',  
'on',  
'time'],  
['exclusive',  
'deal',  
'just',  
'for',  
'you',  
'50',  
'off',  
'on',  
'all',  
'items'],  
['hope', 'you', 'are', 'having', 'a', 'great', 'day'],  
['earn', '500', 'daily', 'from', 'home', 'sign', 'up', 'now'],  
['your', 'subscription', 'has', 'been', 'renewed', 'successfully'],  
['get',  
'rich',  
'quick',  
'invest',  
'in',  
'this',  
'once',  
'in',  
'a',
```

```

'lifetime',
'opportunity'],
['family', 'dinner', 'is', 'planned', 'for', 'this', 'weekend'],
['claim', 'your', 'free', 'cryptocurrency', 'before', 'it', 'expires'],
['see', 'you', 'at', 'the', 'gym', 'tomorrow', 'morning'],
['congratulations',
'you',
'have',
'been',
'selected',
'for',
'a',
'free',
'ipad'],
['the', 'project', 'deadline', 'has', 'been', 'extended'],
['special', 'promotion', 'buy', 'two', 'get', 'one', 'free'],
['can', 'you', 'send', 'me', 'the', 'presentation', 'slides'],
['your', 'paypal', 'account', 'is', 'on', 'hold', 'verify', 'immediately'],
['the', 'weather', 'is', 'great', 'for', 'a', 'picnic', 'today'],
['this',
'is',
'not',
'a',
'drill',
'urgent',
'security',
>alert',
'for',
'your',
'email',
'account'],
['let', 's', 'catch', 'up', 'this', 'weekend'],
['you', 'just', 'received', 'a', '200', 'shopping', 'voucher'],
['lunch', 'at', '1', 'pm', 'today'],
['mega', 'sale', 'starts', 'now', 'up', 'to', '90', 'off'],
['hope', 'you', 're', 'doing', 'well', 'let', 's', 'talk', 'soon'],
['final', 'notice', 'your', 'car', 'warranty', 'is', 'expiring'],
['reminder', 'submit', 'your', 'assignment', 'by', 'friday'],
['you', 'have', 'unclaimed', 'lottery', 'winnings', 'collect', 'now'],
['looking', 'forward', 'to', 'our', 'meeting', 'later'],
['work', 'from', 'home', 'and', 'earn', '10', '000', 'monthly'],
['dinner', 'reservation', 'confirmed', 'for', '8', 'pm'],
['click', 'here', 'to', 'unlock', 'unlimited', 'streaming', 'access'],
['check', 'out', 'this', 'interesting', 'article', 'i', 'found'],
['your', 'loan', 'application', 'has', 'been', 'approved', 'call', 'now'],
['meeting', 'rescheduled', 'to', '2', 'pm'],
['hot', 'singles', 'are', 'waiting', 'for', 'you', 'join', 'now'],
['can', 'you', 'review', 'my', 'report', 'and', 'provide', 'feedback'],
['hurry', 'your', 'exclusive', 'discount', 'code', 'expires', 'soon'],
['thank',
'you',
'for',
'your',
'payment',
'your',
'order',

```

```
'is',
'being',
'processed'],
['you', 'have', 'pending', 'refunds', 'claim', 'now'],
['reminder', 'doctor', 's', 'appointment', 'at', '5', 'pm'],
['this', 'software', 'can', 'help', 'you', 'make', 'money', 'fast'],
['are', 'you', 'available', 'for', 'a', 'quick', 'call'],
['get', 'a', 'free', 'trial', 'of', 'our', 'weight', 'loss', 'program'],
['happy', 'birthday', 'wishing', 'you', 'a', 'fantastic', 'year', 'ahead'],
['upgrade', 'your', 'phone', 'today', 'with', 'our', 'best', 'deals'],
['your', 'package', 'is', 'out', 'for', 'delivery'],
['earn', 'extra', 'cash', 'by', 'taking', 'surveys', 'online'],
['just',
'a',
'friendly',
'reminder',
'to',
'pay',
'your',
'electricity',
'bill'],
['act',
'now',
'your',
'account',
'will',
'be',
'suspended',
'in',
'24',
'hours'],
['looking', 'forward', 'to', 'the', 'weekend', 'getaway'],
['your',
'credit',
'card',
'has',
'been',
'charged',
'contact',
'us',
'for',
'a',
'refund'],
['i', 'll', 'be', 'at', 'the', 'airport', 'at', '6', 'pm'],
['huge', 'giveaway', 'enter', 'now', 'to', 'win', 'amazing', 'prizes'],
['the', 'concert', 'tickets', 'are', 'selling', 'out', 'fast'],
['your',
'netflix',
'account',
'has',
'been',
'compromised',
'reset',
'password',
'now'],
['good', 'luck', 'on', 'your', 'job', 'interview', 'today'],
```

```

['massive',
 'bitcoin',
 'surge',
 'invest',
 'now',
 'before',
 'it',
 's',
 'too',
 'late'],
['meeting', 'with', 'the', 'client', 'has', 'been', 'confirmed'],
['flash', 'sale', 'on', 'premium', 'gadgets', 'limited', 'stock'],
['please', 'find', 'the', 'attached', 'report', 'for', 'your', 'review'],
['warning',
 'unusual',
 'activity',
 'detected',
 'on',
 'your',
 'bank',
 'account'],
['traffic',
 'is',
 'heavy',
 'on',
 'the',
 'way',
 'to',
 'office',
 'leave',
 'early'],
['congratulations',
 'you',
 've',
 'unlocked',
 'a',
 'secret',
 'reward',
 'click',
 'here'],
['how', 'was', 'your', 'weekend', 'let', 's', 'catch', 'up', 'soon']]

```

In [25]: `df["tokens"]`

```

Out[25]: 0      [congratulations, you, ve, won, a, 1000, gift,...
1      [reminder, your, appointment, is, scheduled, f...
2      [urgent, your, account, has, been, compromised...
3      [join, us, for, a, webinar, on, cybersecurity,...
4      [limited, time, offer, buy, one, get, one, fre...
        ...
67     [please, find, the, attached, report, for, you...
68     [warning, unusual, activity, detected, on, you...
69     [traffic, is, heavy, on, the, way, to, office,...
70     [congratulations, you, ve, unlocked, a, secret...
71     [how, was, your, weekend, let, s, catch, up, s...
Name: tokens, Length: 72, dtype: object

```

```
In [7]: word2vec_model = Word2Vec(sentences=corpus, vector_size=100, window=5, min_count=1,
```

```
In [38]: word2vec_model.wv["your"] == word2vec_model.wv[0]
```

```
Out[38]: array([[ True,  True,  True,  True,  True,  True,  True,  True,  True,
                True,  True,  True,  True,  True,  True,  True,  True,  True,
                True,  True,  True,  True,  True,  True,  True,  True,  True,
                True,  True,  True,  True,  True,  True,  True,  True,  True,
                True,  True,  True,  True,  True,  True,  True,  True,  True,
                True,  True,  True,  True,  True,  True,  True,  True,  True,
                True,  True,  True,  True,  True,  True,  True,  True,  True,
                True,  True,  True,  True,  True,  True,  True,  True,  True,
                True,  True,  True,  True,  True,  True,  True,  True,  True,
                True,  True,  True,  True,  True,  True,  True,  True,  True,
                True])
```

```
In [ ]: word2vec_model.wv[0].shape
```

```
Out[ ]: (100,)
```

```
In [35]: word2vec_model.wv.key_to_index
```



```
Out[35]: {'your': 0,  
          'you': 1,  
          'a': 2,  
          'for': 3,  
          'the': 4,  
          'now': 5,  
          'is': 6,  
          'on': 7,  
          'to': 8,  
          'been': 9,  
          'has': 10,  
          'pm': 11,  
          'at': 12,  
          'account': 13,  
          'this': 14,  
          'free': 15,  
          'meeting': 16,  
          'are': 17,  
          's': 18,  
          'get': 19,  
          'just': 20,  
          'in': 21,  
          'out': 22,  
          'our': 23,  
          'today': 24,  
          'weekend': 25,  
          'can': 26,  
          'reminder': 27,  
          'claim': 28,  
          'here': 29,  
          'up': 30,  
          'earn': 31,  
          'have': 32,  
          'click': 33,  
          'it': 34,  
          'be': 35,  
          'soon': 36,  
          'one': 37,  
          'of': 38,  
          'sale': 39,  
          'let': 40,  
          'congratulations': 41,  
          'password': 42,  
          'compromised': 43,  
          'exclusive': 44,  
          'join': 45,  
          'fast': 46,  
          'us': 47,  
          'home': 48,  
          'from': 49,  
          'best': 50,  
          'reward': 51,  
          'limited': 52,  
          'time': 53,  
          'enter': 54,  
          'buy': 55,
```

'great': 56,
'catch': 57,
'all': 58,
'i': 59,
'hope': 60,
'off': 61,
'report': 62,
'please': 63,
'review': 64,
'see': 65,
'how': 66,
're': 67,
'doing': 68,
'reset': 69,
'call': 70,
'year': 71,
'dinner': 72,
'win': 73,
'and': 74,
'by': 75,
'package': 76,
'forward': 77,
'looking': 78,
'appointment': 79,
'invest': 80,
'scheduled': 81,
'quick': 82,
'tomorrow': 83,
'with': 84,
'before': 85,
'confirmed': 86,
'10': 87,
've': 88,
'urgent': 89,
'expires': 90,
'card': 91,
'concert': 92,
'track': 93,
'tickets': 94,
'shipped': 95,
'lucky': 96,
'3': 97,
'vacation': 98,
'netflix': 99,
'selling': 100,
'winner': 101,
'contact': 102,
'prizes': 103,
'deal': 104,
'50': 105,
'cryptocurrency': 106,
'huge': 107,
'planned': 108,
'family': 109,
'opportunity': 110,
'lifetime': 111,

'once': 112,
'giveaway': 113,
'rich': 114,
'successfully': 115,
'renewed': 116,
'subscription': 117,
'amazing': 118,
'sign': 119,
'daily': 120,
'500': 121,
'day': 122,
'having': 123,
'items': 124,
'contest': 125,
'interview': 126,
'iphone': 127,
'detected': 128,
'cybersecurity': 129,
'gadgets': 130,
'webinar': 131,
'stock': 132,
'find': 133,
'attached': 134,
'warning': 135,
'am': 136,
'unusual': 137,
'activity': 138,
'bank': 139,
'offer': 140,
'traffic': 141,
'heavy': 142,
'way': 143,
'office': 144,
'leave': 145,
'gift': 146,
'1000': 147,
'early': 148,
'won': 149,
'unlocked': 150,
'practices': 151,
'premium': 152,
'new': 153,
'biggest': 154,
'brand': 155,
'week': 156,
'next': 157,
'good': 158,
'schedule': 159,
'we': 160,
'luck': 161,
'job': 162,
'morning': 163,
'massive': 164,
'bitcoin': 165,
'flash': 166,
'surge': 167,

'miss': 168,
't': 169,
'don': 170,
'too': 171,
'checking': 172,
'late': 173,
'hello': 174,
'products': 175,
'client': 176,
'gym': 177,
'extended': 178,
'6': 179,
'available': 180,
'make': 181,
'help': 182,
'software': 183,
'5': 184,
'doctor': 185,
'refunds': 186,
'pending': 187,
'processed': 188,
'being': 189,
'order': 190,
'payment': 191,
'thank': 192,
'code': 193,
'discount': 194,
'hurry': 195,
'feedback': 196,
'provide': 197,
'my': 198,
'waiting': 199,
'singles': 200,
'hot': 201,
'2': 202,
'rescheduled': 203,
'approved': 204,
'application': 205,
'loan': 206,
'found': 207,
'money': 208,
'trial': 209,
'interesting': 210,
'weight': 211,
'credit': 212,
'getaway': 213,
'hours': 214,
'24': 215,
'suspended': 216,
'will': 217,
'act': 218,
'bill': 219,
'electricity': 220,
'pay': 221,
'friendly': 222,
'online': 223,

'surveys': 224,
'taking': 225,
'cash': 226,
'extra': 227,
'delivery': 228,
'deals': 229,
'phone': 230,
'upgrade': 231,
'ahead': 232,
'fantastic': 233,
'wishing': 234,
'birthday': 235,
'happy': 236,
'program': 237,
'loss': 238,
'article': 239,
'check': 240,
'selected': 241,
'voucher': 242,
'200': 243,
'received': 244,
'secret': 245,
'll': 246,
'email': 247,
'alert': 248,
'security': 249,
'drill': 250,
'not': 251,
'airport': 252,
'picnic': 253,
'weather': 254,
'immediately': 255,
'verify': 256,
'hold': 257,
'paypal': 258,
'slides': 259,
'presentation': 260,
'me': 261,
'send': 262,
'two': 263,
'promotion': 264,
'special': 265,
'charged': 266,
'deadline': 267,
'project': 268,
'ipad': 269,
'shopping': 270,
'lunch': 271,
'access': 272,
'1': 273,
'streaming': 274,
'unlimited': 275,
'unlock': 276,
'8': 277,
'reservation': 278,
'monthly': 279,

```
'000': 280,  
'work': 281,  
'later': 282,  
'collect': 283,  
'winnings': 284,  
'lottery': 285,  
'unclaimed': 286,  
'friday': 287,  
'assignment': 288,  
'submit': 289,  
'expiring': 290,  
'warranty': 291,  
'car': 292,  
'notice': 293,  
'final': 294,  
'refund': 295,  
'talk': 296,  
'well': 297,  
'90': 298,  
'starts': 299,  
'mega': 300,  
'was': 301}
```

```
In [8]: word_vectors = word2vec_model.wv  
  
vocab = list(word_vectors.key_to_index.keys())  
  
word_embeddings = [word_vectors[word] for word in vocab]
```

```
In [45]: len(word_embeddings)
```

```
Out[45]: 302
```

```
In [40]: for embed in word_embeddings[:10]:  
          print(embed, embed.shape)
```

```
[-0.00067511  0.000554    0.00516026  0.00899018 -0.00939332 -0.00761226
 0.00661836  0.00968759 -0.00510594 -0.00396155  0.00731074 -0.00189977
-0.00473711  0.00660262 -0.00488015 -0.00221246  0.00281851  0.00077514
-0.00846202 -0.00995755  0.00751478  0.0049938  0.00708692  0.00062938
 0.00627045 -0.00320369 -0.00111669  0.00553427 -0.00793706 -0.0039202
-0.00728954 -0.00082704  0.00973216 -0.00744624 -0.00247191 -0.00157475
 0.00814332 -0.00617375 -0.00026427 -0.0053865  -0.00952247  0.0046833
-0.00879667 -0.00441946  0.0002799  -0.00075133 -0.00788194  0.00940965
 0.00529046  0.00920209 -0.00814662  0.00422575 -0.00413915  0.00099014
 0.0083956  -0.00421544  0.00467295 -0.00669106 -0.00387153  0.00935947
-0.00147233  0.00043624 -0.00398405 -0.00783146 -0.00196044  0.00251512
-0.00043383  0.00613491 -0.00328447  0.0026424  0.00552137  0.00841064
-0.00127188 -0.00904392  0.00475507  0.00117123  0.00749201 -0.00075442
-0.00311807 -0.00854016 -0.00097858  0.00282458  0.00509274  0.0074435
-0.00592162  0.00201272  0.00621266 -0.00424708 -0.00273102  0.00680019
 0.00220472  0.00041515  0.00335278  0.00011503  0.00999404  0.00571929
-0.00853464 -0.0073861  0.00099111  0.00611237] (100,)
[-8.67178198e-03  3.80311301e-03  5.20101981e-03  5.71057992e-03
 7.43043702e-03 -6.40069554e-03  1.13846629e-03  6.38654456e-03
-2.91368179e-03 -6.25029579e-03 -4.26045066e-04 -8.53904616e-03
-5.66052739e-03  7.10265897e-03  3.38770240e-03  7.02390540e-03
 6.76770415e-03  7.40948040e-03 -3.86121939e-03 -8.40064837e-04
 2.42219958e-03 -4.53365734e-03  8.50223284e-03 -9.92696267e-03
 6.74183248e-03  2.97904550e-03 -4.99278679e-03  4.26755287e-03
-1.90257456e-03  6.73556747e-03  1.00872135e-02 -4.31895629e-03
-5.23661263e-04 -5.79104759e-03  3.77525133e-03  2.95674149e-03
 6.90427795e-03  5.97084966e-03  9.42358188e-03  8.99447128e-03
 7.93163572e-03 -7.15332013e-03 -9.16899554e-03 -2.96486687e-04
-2.94495770e-03  7.73128821e-03  5.80769917e-03 -1.62434718e-03
 1.63892529e-03  1.78700965e-03  7.83835724e-03 -9.58318170e-03
-1.87257072e-04  3.52606387e-03 -1.00156094e-03  8.46248586e-03
 9.06905532e-03  6.57108054e-03 -8.76744627e-04  7.72681227e-03
-8.49904492e-03  3.27665941e-03 -4.58353106e-03 -5.17849391e-03
 3.38164135e-03  5.40782092e-03  7.96544086e-03 -5.50270965e-03
 7.16736214e-03  6.80247927e-03 -3.75301158e-03 -8.71128775e-03
 5.50324237e-03  6.57941541e-03 -6.11798780e-04 -6.47212518e-03
-7.03306170e-03 -2.48595467e-03  4.93531721e-03 -3.55132553e-03
-9.40680504e-03  3.84235848e-03  4.73321509e-03 -6.24897471e-03
 1.12507888e-03 -2.00149138e-03  6.42071755e-05 -9.65860765e-03
 2.86586303e-03 -4.74060653e-03  1.35124195e-03 -1.45551597e-03
 2.11918983e-03 -7.90429115e-03 -2.51940475e-03  2.95513356e-03
 5.55295777e-03 -2.59759324e-03 -9.46085900e-03  4.36392426e-03] (100,)
[ 1.0636411e-05  3.1937335e-03 -6.7833830e-03 -1.3589377e-03
 7.5925947e-03  7.1195923e-03 -3.5968269e-03  2.9639611e-03
-8.3619766e-03  6.1341515e-03 -4.6456219e-03 -3.3853375e-03
 9.1700898e-03  8.8338403e-04  7.4509182e-03 -6.2390869e-03
 5.1013459e-03  9.7850598e-03 -8.5234269e-03 -5.3579872e-03
-6.9744703e-03 -4.8855464e-03 -3.6378691e-03 -8.6057764e-03
 7.8915637e-03 -4.7830553e-03  8.3846310e-03  5.0868266e-03
-6.7235422e-03  4.0165386e-03  5.5770581e-03 -7.4033206e-03
-7.3389201e-03 -2.5250174e-03 -8.6457115e-03 -1.3654133e-03
-3.2138536e-04  3.1353272e-03  1.2934080e-03 -1.1413393e-03
-5.4978747e-03  1.5455949e-03 -8.7404158e-04  6.8067634e-03
 4.1490369e-03  4.3669050e-03  1.3300749e-03 -2.8088058e-03
-4.2186221e-03 -1.0684946e-03  1.5334625e-03 -2.7007568e-03
-7.0183706e-03 -7.7365553e-03 -9.1720317e-03 -5.8582653e-03
```

```
-1.8068694e-03 -4.2629265e-03 -6.6471533e-03 -3.7058906e-03
4.2957198e-03 -3.6515731e-03 8.4651988e-03 1.4099468e-03
-7.4456506e-03 9.4195465e-03 7.7770599e-03 5.8093364e-03
-7.0556067e-03 6.0212361e-03 4.0171216e-03 5.2147396e-03
4.2768461e-03 2.0356267e-03 -2.9677285e-03 8.6488323e-03
9.6583869e-03 3.8293772e-03 -3.0685673e-03 1.3397112e-04
1.1834587e-03 -8.4653497e-03 -8.3335731e-03 -2.8868124e-06
1.0875711e-03 -5.6255222e-03 -4.6878792e-03 -7.1153170e-03
8.4952721e-03 1.5492617e-04 -4.2949701e-03 5.8622747e-03
9.0762926e-03 -4.1181319e-03 8.1298603e-03 5.7006758e-03
6.0631218e-03 3.5792886e-04 8.2463715e-03 -7.1396520e-03] (100,)
[-8.2676737e-03 9.3952203e-03 -1.8432213e-04 -1.9774267e-03
4.5751221e-03 -4.3015713e-03 2.7873819e-03 7.1886936e-03
6.0047405e-03 -7.5585525e-03 9.3355561e-03 4.5200381e-03
3.8628692e-03 -6.1933231e-03 8.4566046e-03 -2.3257406e-03
8.7640733e-03 -5.4387143e-03 -8.1720874e-03 6.6334377e-03
1.7578781e-03 -2.2117079e-03 9.5954081e-03 9.4428239e-03
-9.8235635e-03 2.5135598e-03 6.0949200e-03 3.7531340e-03
1.8894948e-03 4.4771284e-04 7.6227268e-04 -3.7875790e-03
-7.0832251e-03 -2.1289962e-03 3.8779953e-03 8.9802258e-03
9.2446757e-03 -6.0832524e-03 -9.4992416e-03 9.5448531e-03
3.4608960e-03 5.0407238e-03 6.2917056e-03 -2.7472274e-03
7.4521843e-03 2.7017442e-03 2.8047236e-03 -2.4441660e-03
-3.0329784e-03 -2.3629807e-03 4.3065231e-03 3.8552265e-05
-9.5442710e-03 -9.6644117e-03 -6.1542480e-03 -5.1423889e-05
2.0624239e-03 9.4920779e-03 5.4978877e-03 -4.2774887e-03
3.1178346e-04 5.0289943e-03 7.7399183e-03 -1.1892678e-03
4.1503864e-03 -5.8250460e-03 -6.8063091e-04 8.3294753e-03
-2.5647364e-03 -9.4979033e-03 5.7741967e-03 -3.9237095e-03
-1.1734820e-03 1.0033232e-02 -2.0528492e-03 -4.5450213e-03
-5.2672247e-03 7.0113679e-03 -5.8977469e-03 2.2062084e-03
-5.2942829e-03 6.1262357e-03 4.2151771e-03 2.7668667e-03
-1.5663444e-03 -2.6851792e-03 9.0226382e-03 5.4155863e-03
-2.0054805e-03 -9.4512803e-03 -7.2556804e-03 -9.7680860e-04
-8.1385009e-04 -2.5550292e-03 9.7981496e-03 -2.1485287e-04
6.0327239e-03 -7.5914157e-03 -2.4644556e-03 -5.6709582e-03] (100,)
[-0.00715185 0.00137409 -0.00719281 -0.00224806 0.00365651 0.00566329
0.00125135 0.00236436 -0.00418165 0.00716532 -0.00633462 0.00455538
-0.00831883 0.00206636 -0.00496462 -0.00430106 -0.00314938 0.00562588
0.00575901 -0.00513692 0.0008179 -0.00852805 0.00794492 0.00921341
-0.00275266 0.00087567 0.00069543 0.00535359 -0.00869045 0.00062316
0.0069356 0.00225877 0.00119791 -0.00936299 0.0084448 -0.00612467
-0.00294852 0.00335863 -0.00083343 0.00120713 0.00183498 -0.00694058
-0.00971514 0.00905237 0.0062629 -0.00704404 0.00333054 0.00015946
0.00486047 -0.00714697 0.00407268 0.00427287 0.00997922 -0.00448268
-0.00140103 -0.00725971 -0.00966199 -0.00903383 -0.00112002 -0.00649336
0.00487396 -0.00611601 0.00259826 0.0007166 -0.00353393 -0.00097024
0.01010558 0.00931911 -0.00465017 0.00920154 -0.00564357 0.00596308
-0.00308154 0.00347546 0.00317189 0.00709731 -0.0023416 0.00878435
0.0074569 -0.00948472 -0.00803742 -0.00764935 0.00281861 -0.00261517
-0.00704632 -0.0080832 0.00838264 0.00218216 -0.00919433 -0.00481466
0.00329905 -0.00458374 0.00524483 -0.0042368 0.00277745 -0.00788058
0.00631764 0.00467473 0.00085252 0.00291277] (100,)
[-8.7691620e-03 2.2519329e-03 -8.5430004e-04 -9.3522584e-03
-9.4880071e-03 -1.5777372e-03 4.4966154e-03 3.9842329e-03
-6.5228343e-03 -6.9281259e-03 -4.9947067e-03 -2.4080924e-03
```



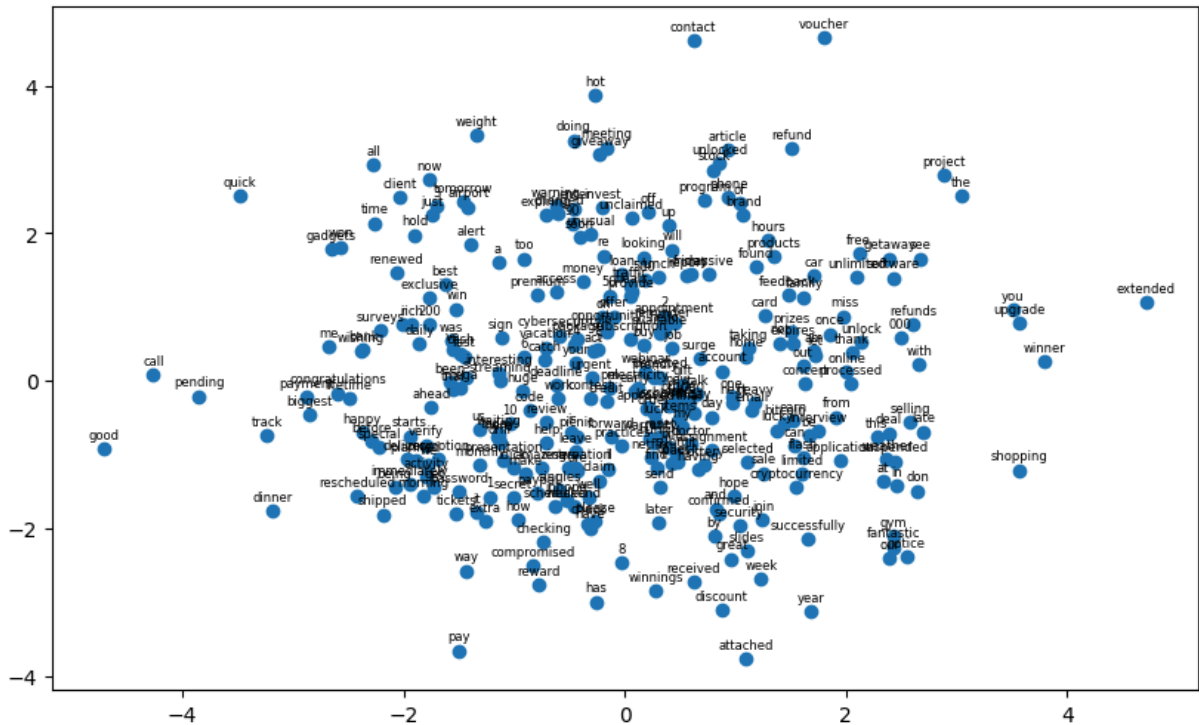
```
-7.2663282e-03 -9.5857130e-03 -2.7366797e-03 -8.4862178e-03
-6.0904142e-03 -5.7519469e-03 -2.3557676e-03 -1.8864083e-03
-8.8795256e-03 -8.1260246e-04 8.2458928e-03 7.6532401e-03
-7.2442256e-03 -3.6249061e-03 3.0930014e-03 -9.6971737e-03
1.3406293e-03 6.4977957e-03 5.8083804e-03 -8.7097576e-03
-4.4646794e-03 -8.1826029e-03 8.9780435e-05 9.4393399e-03
6.0143825e-03 4.9630702e-03 4.9369070e-03 -3.4676285e-03
9.5944842e-03 -7.4635618e-03 -7.2501381e-03 -2.2715128e-03
-6.6218601e-04 -3.3503054e-03 -6.8206881e-04 7.4425372e-03
-6.1979599e-04 -1.6632844e-03 2.7807818e-03 -8.4184697e-03
7.8208540e-03 8.5886065e-03 -9.6128508e-03 2.4989790e-03
9.9662729e-03 -7.6212408e-03 -7.1034525e-03 -7.7459184e-03
8.4526297e-03 -6.1576685e-04 9.2537897e-03 -8.1834532e-03
3.5820804e-03 2.6298056e-03 9.0650277e-04 2.5372377e-03
-7.6437406e-03 -9.1752531e-03 2.3801981e-03 6.1775944e-03
8.0176974e-03 5.8190241e-03 -6.0439133e-04 8.5272156e-03
-9.3240533e-03 3.4253278e-03 9.5059288e-05 3.9532152e-03
7.3488867e-03 -6.7270622e-03 5.4941778e-03 -9.3699722e-03
-9.3206903e-04 -8.6281355e-03 -5.0535905e-03 9.4710793e-03
-1.7020634e-03 2.9043141e-03 9.2328247e-03 9.0234745e-03
-8.2670888e-03 -3.0260610e-03 1.0033353e-02 5.3254524e-03
-1.4636738e-03 -8.8694878e-03 2.9841424e-03 -6.8093524e-03] (100,)
[ 8.0458671e-03 -4.3259142e-03 -1.0517903e-03 1.0308541e-03
-2.1443210e-04 8.9606689e-04 6.2052230e-03 3.5003119e-04
-3.2918293e-03 -1.5939729e-03 5.9076198e-03 1.3243263e-03
-8.3238760e-04 9.3894089e-03 -4.9100933e-03 -1.0420205e-03
9.1516133e-03 6.6716666e-03 1.4021215e-03 -9.1326600e-03
1.2377385e-03 -2.2942924e-03 9.5412415e-03 1.1818863e-03
1.4570171e-03 2.4868783e-03 -1.8639708e-03 -5.0793011e-03
8.7545865e-05 -2.0015284e-03 6.6288929e-03 8.9678569e-03
-6.3782436e-04 2.9456059e-03 -6.1546708e-03 1.8825206e-03
-6.8937093e-03 -8.8808052e-03 -6.0287169e-03 -9.2140967e-03
7.3616127e-03 -5.8535836e-03 8.3176866e-03 -7.2468081e-03
3.6223326e-03 9.5056342e-03 -7.8772996e-03 -1.0021881e-02
-4.1743401e-03 -2.7134554e-03 -2.0175106e-04 -8.9307670e-03
-8.6118747e-03 2.7915458e-03 -8.2361121e-03 -8.9794258e-03
-2.2807091e-03 -8.5223792e-03 -7.2115925e-03 -8.3529474e-03
-2.5519656e-04 -4.5372560e-03 6.7396872e-03 1.4769792e-03
-3.5616076e-03 6.1340248e-03 -5.8718799e-03 -4.3468075e-03
-7.4790460e-03 -4.1520051e-03 -1.8055133e-03 6.5592080e-03
-2.6841371e-03 4.9939547e-03 7.0979828e-03 -7.1860421e-03
4.6198568e-03 6.1906236e-03 -3.2020621e-03 6.7513855e-03
6.0804118e-03 -6.4167669e-03 -6.9021345e-03 2.7340183e-03
-1.7347807e-03 -5.9642438e-03 9.6036121e-03 -4.9162097e-03
-6.3658282e-03 -1.2185374e-04 -2.4986491e-03 5.7982933e-04
-3.5727273e-03 -4.2266626e-04 -4.6816439e-04 1.1244711e-03
8.3468473e-03 -5.9106438e-03 -1.5926715e-03 5.4942067e-03] (100,)
[ 8.10617115e-03 -4.29666927e-03 9.01538506e-03 8.28506239e-03
-4.52621328e-03 4.33454952e-05 4.30895900e-03 -3.56600131e-03
-5.64286439e-03 -6.59653917e-03 -6.92570175e-04 -4.72592859e-04
4.36380738e-03 -2.45878659e-03 -1.63355813e-04 2.23836722e-03
4.81842505e-03 -1.88027494e-04 -6.39549782e-03 -9.51041933e-03
1.08188186e-04 6.63757417e-03 1.63554680e-03 -9.04877670e-03
-7.96560943e-03 6.64540241e-03 -3.86866299e-03 6.10653544e-03
-6.79506827e-03 8.50493275e-03 -6.41362416e-03 3.34705110e-03
-9.36501427e-04 -6.84938440e-03 -3.35866748e-03 -9.48399713e-04
```

-5.46221575e-03 -1.34552759e-03 -7.72235729e-03 2.30694562e-03
9.08800308e-03 -2.51845550e-03 -9.48754780e-04 3.52569879e-03
8.77651572e-03 -6.09844550e-03 -6.99522486e-03 -3.02654784e-03
9.29127261e-03 8.14937172e-04 -8.63888860e-03 -1.52539834e-03
9.46105551e-03 -7.51377828e-03 -5.44090662e-03 9.44081135e-03
-8.91222339e-03 3.85252433e-03 4.71033243e-04 6.65077707e-03
8.32540356e-03 -2.74057128e-03 -3.96231143e-03 8.87887646e-03
1.84573408e-03 6.26133569e-03 -9.26426239e-03 9.85514279e-03
-1.64101634e-03 -5.82334585e-03 2.96191382e-03 -4.05515428e-04
4.78030322e-03 -2.21226527e-03 -3.90456314e-03 2.56513478e-03
8.39749537e-03 -4.98343538e-03 2.43528257e-03 -7.84468185e-03
-6.80864602e-03 -4.69512539e-04 -8.92369077e-03 3.00200470e-03
1.46080868e-03 -2.24934309e-03 5.07021463e-03 1.00143990e-02
8.66542570e-03 -1.90408982e-03 2.31134333e-03 -3.86109040e-03
-8.30879528e-03 6.25716010e-03 -1.76520599e-03 -3.80397338e-04
-1.58044568e-03 -4.71622031e-03 4.10648296e-03 -4.41476889e-03] (100,)
[-9.6211694e-03 9.1404887e-03 4.2044353e-03 9.1902195e-03
6.5520951e-03 2.6453400e-03 9.9274153e-03 -4.0845005e-03
-6.8910997e-03 4.1111684e-03 3.7849962e-03 -5.8311378e-03
9.6484665e-03 -3.5234967e-03 9.5684780e-03 6.5607083e-04
-6.3444511e-03 -2.0498238e-03 -7.4735587e-03 -3.2485700e-03
1.1249057e-03 9.4289044e-03 9.5001506e-03 -6.6836132e-03
3.4322650e-03 2.3777033e-03 -2.6096927e-03 -9.4051203e-03
8.8725047e-04 -8.1902361e-03 6.4578452e-03 -5.7107578e-03
5.6144218e-03 9.7055044e-03 -1.4884614e-04 4.6408996e-03
-1.6955107e-03 7.2554178e-03 3.7843531e-03 -9.3534617e-03
-2.3656597e-03 3.4967060e-03 -6.8616726e-05 -1.2225811e-03
-9.6314342e-04 -1.8711343e-03 5.1640137e-04 4.1271602e-03
-4.0984941e-03 -3.8003626e-03 -1.1001528e-05 1.5331943e-04
-1.9776142e-04 -4.6921298e-03 4.2597433e-03 -2.0476556e-03
2.1861917e-03 7.1084005e-04 5.7543684e-03 -6.8920655e-03
-6.7228992e-03 -4.4200630e-03 9.5818611e-03 -1.6494561e-03
-9.6519710e-03 -4.8995746e-04 -4.2051473e-03 6.2307226e-03
-9.8917577e-03 3.0761322e-03 -9.2631523e-03 1.3007466e-03
6.0463753e-03 7.4515552e-03 -7.3908060e-03 -5.7979114e-03
-6.8141245e-03 -7.8702671e-03 -9.7611807e-03 -2.0418263e-03
-8.4432703e-04 -7.2273095e-03 6.6454345e-03 1.3211168e-03
5.6855539e-03 1.5239783e-03 8.5963565e-04 -7.1280277e-03
-2.0221602e-03 4.3394719e-03 -4.8352513e-03 1.2666074e-03
2.8176452e-03 -1.5354153e-03 1.0197141e-02 8.6447848e-03
2.5408354e-03 6.9818269e-03 5.9047532e-03 -5.6869802e-03] (100,)
[-0.00517515 -0.00656754 -0.00774546 0.0082966 -0.00202189 -0.00698988
-0.00410234 0.0053617 -0.00291415 -0.00378364 0.00165329 -0.0028819
-0.00163084 0.00105448 -0.00299346 0.00842175 0.00388618 -0.01000396
0.00621315 -0.00692917 0.0008306 0.00436354 -0.00502731 -0.00215658
0.0080977 -0.00418723 -0.00768465 0.00915999 -0.00221242 -0.00472156
0.00862575 0.0043327 0.00444231 0.00925403 -0.00848523 0.00531542
0.0020745 0.00406893 0.0016619 0.00425617 0.00453872 0.00599189
-0.00319702 -0.00458707 -0.00035867 0.00245474 -0.00338546 0.00602519
0.00424066 0.00775408 0.00259309 0.00806689 -0.00137152 0.00812564
0.00368743 -0.008006 -0.00391553 -0.00244636 0.00477977 -0.0008903
-0.00282405 0.00786081 0.00937741 -0.00168467 -0.00528502 -0.00468844
-0.00470895 -0.00945946 0.00117458 -0.00413522 0.0024933 0.00562167
-0.00403316 -0.00952258 0.00164587 -0.0065943 0.00251808 -0.00375492
0.00694847 0.00071135 0.00351713 -0.00271796 -0.00176059 0.00780147
0.0013273 -0.00580479 -0.00778803 0.00137236 0.00657107 0.00558083

```
-0.00881656 0.00872005 0.00401507 0.00747026 0.00988504 -0.00711224
-0.00899974 0.00573431 0.00937539 0.00347604] (100,)
```

```
In [ ]: pca = PCA(n_components=2)
reduced_embeddings = pca.fit_transform(StandardScaler().fit_transform(word_embeddings))

plt.figure(figsize=(10, 6))
plt.scatter(reduced_embeddings[:, 0], reduced_embeddings[:, 1], marker='o')
for i, word in enumerate(vocab): # `words` is the list of words corresponding to t
    plt.annotate(word, (reduced_embeddings[i, 0], reduced_embeddings[i, 1]),
                  textcoords="offset points", xytext=(0, 5), ha='center', fontsize=6)
plt.show()
```



```
In [56]: # reduced_embeddings[1] == reduced_embeddings[word2vec_model.wv.key_to_index.get(1)]
```

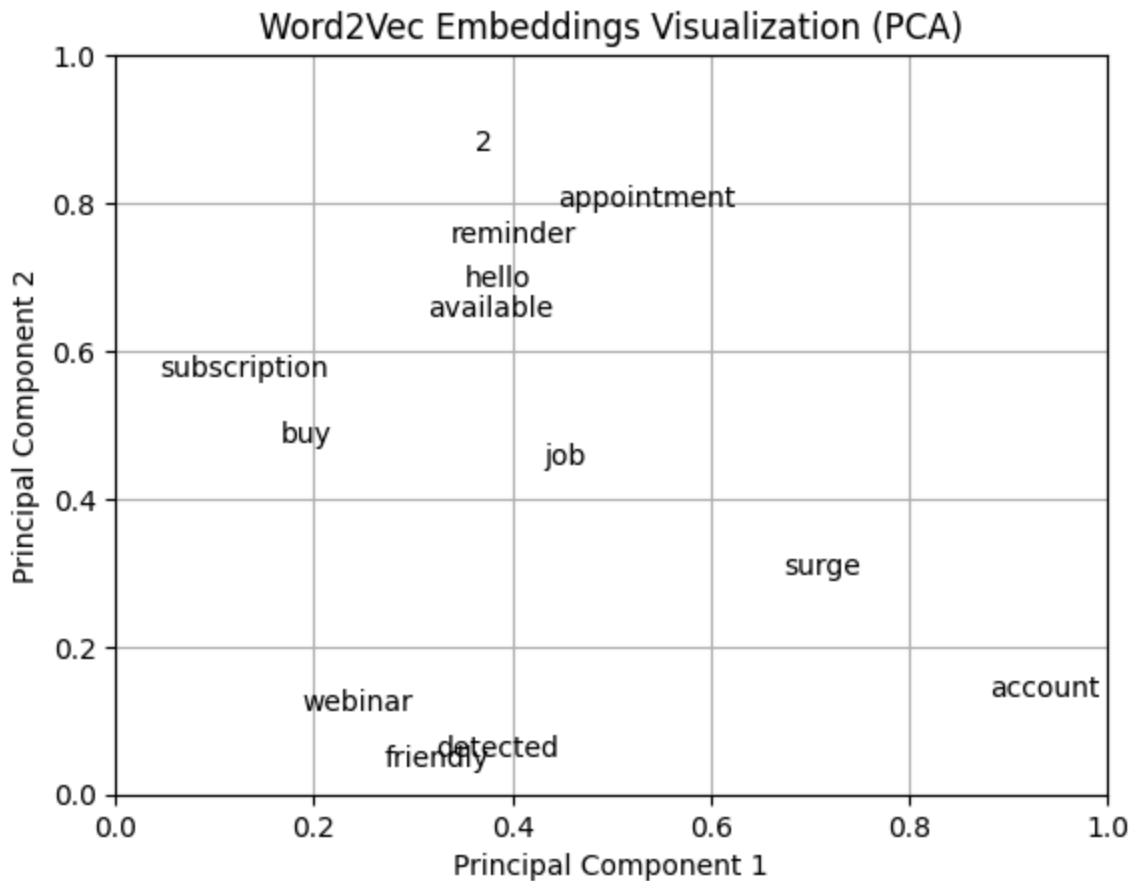
```
In [12]: reduced_embeddings.shape
```

```
Out[12]: (302, 2)
```

```
In [ ]: for i, word in enumerate(vocab):
    plt.annotate(word, xy=(reduced_embeddings[i, 0], reduced_embeddings[i, 1]))

plt.title("Word2Vec Embeddings Visualization (PCA)")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.grid(True)
plt.show()

print("Most similar words to 'meeting':", word_vectors.most_similar('meeting'))
```



Most similar words to 'meeting': [('thank', 0.25929632782936096), ('will', 0.24854585528373718), ('access', 0.23469451069831848), ('enter', 0.23448140919208527), ('you r', 0.22572541236877441), ('ve', 0.2227151244878769), ('money', 0.21266399323940277), ('looking', 0.1991436928510666), ('luck', 0.19419558346271515), ('2', 0.18915194272994995)]

```
In [57]: print("Most similar words to 'meeting':", word_vectors.most_similar('offer'))
```

Most similar words to 'meeting': [('lucky', 0.30802595615386963), ('brand', 0.25161880254745483), ('renewed', 0.2469548135995865), ('lunch', 0.24616022408008575), ('time', 0.22845079004764557), ('subscription', 0.22336992621421814), ('send', 0.21646657586097717), ('practices', 0.19851689040660858), ('late', 0.19264769554138184), ('great', 0.1922316551208496)]

```
In [59]: for i, word in enumerate(vocab):
          print(i, word, (reduced_embeddings[i, 0], reduced_embeddings[i, 1]))
          # plt.annotate(word, xy=(reduced_embeddings[i, 0], reduced_embeddings[i, 1]))
```

0 your (-0.4398754095036703, 0.23480449820057925)
1 you (3.5078891023800765, 0.9589720710793257)
2 a (-1.1464715950268443, 1.6084549640201518)
3 for (0.46381893083866493, -0.32996611125048464)
4 the (3.048319093510981, 2.511062185554566)
5 now (-1.7682208307966891, 2.7252654258927254)
6 is (-0.4788632805447565, -0.6956007346728704)
7 on (-0.1983950795645411, 0.8633923525003783)
8 to (-0.1624306327275243, 0.67045497055479)
9 been (-1.584762093684212, -0.012833896842902004)
10 has (-0.2643867267762733, -2.9914290271216744)
11 pm (-0.141132325228075, -0.10433362657845323)
12 at (2.330366642851629, -1.365080865936816)
13 account (0.8809343875734975, 0.1324738699181384)
14 this (2.277761891542032, -0.7571602497362704)
15 free (2.1187939237095144, 1.7263865888699963)
16 meeting (-0.16237602318795769, 3.156489013185124)
17 are (-0.4316882105846144, -1.215849953733034)
18 s (-1.4336468538659384, 0.04294652955159961)
19 get (-1.7226585252386444, -1.4454704153296427)
20 just (-1.748158371338592, 2.238043318912005)
21 in (2.46059639705019, -1.4217527362667957)
22 out (1.6105409174998917, 0.20744778584041754)
23 our (2.3953395617868893, -2.4012320305320993)
24 today (-1.149918377500998, -0.7547622770225454)
25 weekend (-0.4614384366383481, -1.6940724300225596)
26 can (1.5385914109729237, -0.8760252691969281)
27 reminder (0.33794997517017483, 0.7457384279887302)
28 claim (-0.23148776411262328, -1.3501357459300352)
29 here (-1.1373548858764548, -0.786201875229373)
30 up (0.3962244132654926, 2.105525501067171)
31 earn (1.5212950938640235, -0.5391978978189376)
32 have (-0.3099379436466317, -1.991009424691815)
33 click (-1.0116850134766098, -1.1788797313610349)
34 it (-1.3415770225901182, -1.7751785745941553)
35 be (1.6635885061192697, -0.7373065056045894)
36 soon (-0.4065794073278699, 1.9496307163883573)
37 one (0.970173501941812, -0.21140998320377294)
38 of (1.0334640492800213, 2.4118879864965668)
39 sale (1.2580384215292328, -1.251621418006096)
40 let (1.7180652216498307, 0.3503264805249568)
41 congratulations (-2.592398945648612, -0.16645894666704625)
42 password (-1.5042397314243066, -1.497289221792287)
43 compromised (-0.8318841592704679, -2.502397165119256)
44 exclusive (-1.7669317356527698, 1.1266514948332795)
45 join (1.2451339147902332, -1.8881095877051366)
46 fast (-1.4487556124403118, 0.332942924228124)
47 us (-1.3194533646589035, -0.6583148718529958)
48 home (1.0986653279496468, 0.33261302643699614)
49 from (1.912989546219819, -0.4888981265563153)
50 best (-1.6270104629933908, 1.3002929261114513)
51 reward (-0.7825910539107099, -2.766862597971749)
52 limited (1.604818261499046, -1.2637377571844672)
53 time (-2.2635731485106967, 2.1333894869707324)
54 enter (-0.45309125003580636, 2.337358994975643)
55 buy (0.16710608762081905, 0.47608031261507655)

56 great (0.9618414320500523, -2.4100788412951375)
57 catch (-0.7320588745484722, 0.2790698156571288)
58 all (-2.2736258545913324, 2.92645157632901)
59 i (-2.0126706315835774, 0.7664738714616224)
60 hope (0.981894377159387, -1.5675721619177494)
61 off (0.21533667228042283, 2.28355285471465)
62 report (0.5573337131334458, 1.4192398769973533)
63 please (-0.270625792236978, -1.8953239454730857)
64 review (-0.7145130004389345, -0.5661209475930639)
65 see (2.6657258305825806, 1.656365203138465)
66 how (-0.9629450281243548, -1.8850897206119648)
67 re (-0.19341943407340903, 1.6940958012963878)
68 doing (-0.4649160030507284, 3.2565305044179413)
69 reset (0.31689945648379636, -0.7661200710810521)
70 call (-4.265827016185819, 0.07519164311026252)
71 year (1.6838204270235875, -3.1150719967836)
72 dinner (-3.190972557432254, -1.755648369756222)
73 win (-1.5216812811666884, 0.9753300943417716)
74 and (0.8233683310421631, -1.7316184735689615)
75 by (0.8070348121051392, -2.1012455614206575)
76 package (-0.4365555025422262, 0.5672512013375317)
77 forward (-0.12034212868587064, -0.7657748205355508)
78 looking (0.16899879527713113, 1.6723123317391717)
79 appointment (0.44517237543924676, 0.7946913424451068)
80 invest (-0.20847399896506325, 2.3379617730500195)
81 scheduled (-0.6270878360360495, -1.7046486001947216)
82 quick (-3.4816558896733962, 2.5110130001354563)
83 tomorrow (-1.456572538738719, 2.42493743264845)
84 with (2.6606660412502, 0.22737388735109543)
85 before (-2.292845595175716, -0.8124848746971802)
86 confirmed (0.8503562742695211, -1.7971912149196392)
87 10 (-1.0392911056061125, -0.573366322800764)
88 ve (-1.551465980685323, 0.41740186754719194)
89 urgent (-0.3013059591294818, 0.034658430178712965)
90 expires (1.5235386532523192, 0.5002624253429789)
91 card (1.2602520397027752, 0.8832256512773091)
92 concert (1.6267591090461029, -0.03261303257525308)
93 track (-3.235398040209106, -0.7435561060546715)
94 tickets (-1.5234622186584472, -1.8067757635788324)
95 shipped (-2.188331985196562, -1.8091198506528545)
96 lucky (1.3788326169465892, -0.6698600907305603)
97 3 (-1.6993006483813302, 2.3631203062022204)
98 vacation (-0.7178218200004336, 0.4514632689007835)
99 netflix (0.23637087191113745, -1.0369446061663599)
100 selling (2.578751747690478, -0.5522065896800484)
101 winner (3.800364739617803, 0.2649979269760443)
102 contact (0.6193520524773071, 4.6015714286065625)
103 prizes (1.5064648108997496, 0.6672093435253487)
104 deal (2.3863983679725913, -0.7079095268303851)
105 50 (-0.14117881677945654, 1.1536929234156397)
106 cryptocurrency (1.5486506665364586, -1.4393073416543545)
107 huge (-0.9297272613913748, -0.13580547595257153)
108 planned (-1.891704340209693, -1.0832857862636716)
109 family (1.6198225624067917, 1.1331479587802722)
110 opportunity (-0.16044831207063998, 0.700219416609865)
111 lifetime (-2.4956162287588493, -0.23072161867226806)

112 once (1.848707901108627, 0.6339874412453549)
113 giveaway (-0.22851268232475308, 3.0642720225231517)
114 rich (-1.924193306080346, 0.7631302522319602)
115 successfully (1.6561060041260214, -2.1460140676432466)
116 renewed (-2.065632920106743, 1.4619522524133153)
117 subscription (0.04501240145282743, 0.5663680336368951)
118 amazing (-0.758412417675931, -1.1724400008777853)
119 sign (-1.1195963645799085, 0.5757426396365424)
120 daily (-1.8603803137799966, 0.505293851484455)
121 500 (0.1772076920041587, 1.363661693918258)
122 day (0.7898665269253172, -0.4906458713044544)
123 having (0.6622986354887815, -1.2052863265418519)
124 items (0.4918451510138037, -0.5199137882704098)
125 contest (-0.30856522492703026, -0.23489747059649632)
126 interview (1.750741147610405, -0.6820814713812314)
127 iphone (-0.5303083660707828, -1.6255403853523414)
128 detected (0.322870225855267, 0.0523079668726633)
129 cybersecurity (-0.5884742147648641, 0.6131208826039685)
130 gadgets (-2.6555088282298542, 1.7783532134028448)
131 webinar (0.18749347418478685, 0.114402406150678)
132 stock (0.8025804671294215, 2.8566424720824495)
133 find (0.28974702522192053, -1.1854483390974837)
134 attached (1.0923798164986671, -3.7669271966272158)
135 warning (-0.6191292884979831, 2.370526071971638)
136 am (1.711922486156233, 0.3997152348511858)
137 unusual (-0.3078660194370803, 1.984219096719171)
138 activity (-1.7995206472943241, -1.2989516773681076)
139 bank (-2.3689288589655555, 0.4194174784148957)
140 offer (-0.09889699845964074, 0.8713564411784674)
141 traffic (0.05454759182002357, 1.289305071581121)
142 heavy (1.1710457198350788, -0.2929144891564861)
143 way (-1.431704147503757, -2.5736365965895605)
144 office (0.47789166811355993, -0.2528194015697094)
145 leave (-0.4442623219598671, -0.9575313574608958)
146 gift (0.5235304146174524, -0.049240779240380554)
147 1000 (0.5786391198011288, -0.2813230455559052)
148 early (0.09194031507196829, -0.1366163909738465)
149 won (-2.570510961341941, 1.8140918526639744)
150 unlocked (0.8505273051727889, 2.9486312012329385)
151 practices (-0.023829636632930695, -0.8865755352610877)
152 premium (-0.7890358484787199, 1.1655343078445977)
153 new (0.4775967252421292, -0.1673184238120236)
154 biggest (-2.857508272941001, -0.4616245307200495)
155 brand (1.0706935537259883, 2.247506835192772)
156 week (1.2315645012808618, -2.6832993482396974)
157 next (0.9789391061760901, -0.29213554174581263)
158 good (-4.706886294775726, -0.9140603991509637)
159 schedule (0.39458047809044927, -0.34494431657140345)
160 we (-1.7768104645887859, -1.1302202261732566)
161 luck (0.27270024809150684, -0.5648086321186946)
162 job (0.43094044307058305, 0.44582957066186474)
163 morning (-1.8218908558186755, -1.552877361568644)
164 massive (0.7543709035370482, 1.451143578678208)
165 bitcoin (1.4416375247354072, -0.5852296363437792)
166 flash (1.612674140414257, -1.0394355713474979)
167 surge (0.6729402306805562, 0.2979308771988325)

168 miss (1.9821226733356752, 0.8752576023472064)
169 t (0.6297230380770141, -0.43919779633796985)
170 don (2.6453058496780257, -1.4920031791759383)
171 too (-0.910957590551958, 1.653284119088295)
172 checking (-0.7338671099779789, -2.178763749923066)
173 late (2.696557177768, -0.690581252029335)
174 hello (0.3527596387059586, 0.6875412759690156)
175 products (1.3409210941512417, 1.6845716189662483)
176 client (-2.030290466473569, 2.493224502464213)
177 gym (2.424489556649975, -2.104825971946986)
178 extended (4.715582366462536, 1.0573223170277717)
179 6 (-0.9096489479807796, 0.3226686820801866)
180 available (0.31491813963813564, 0.6470204492601614)
181 make (-0.9031537996046316, -1.2640439753231927)
182 help (-0.7125162737398595, -0.8322635875898955)
183 software (2.4300514534713984, 1.3952736281044218)
184 5 (-0.24053795711629034, 0.42836205303995134)
185 doctor (0.5995125325540386, -0.856733745174667)
186 refunds (2.6092994519703003, 0.7706529855449857)
187 pending (-3.852354407741541, -0.21001489702307136)
188 processed (2.0388051802840415, -0.03668265934542877)
189 being (-2.0830608468928413, -1.4436379930558643)
190 order (0.2660735057456037, -0.43383192279632565)
191 payment (-2.87694186785646, -0.22229869200012942)
192 thank (2.0558294207335384, 0.377143781644002)
193 code (-0.8574850732212735, -0.39844586060596504)
194 discount (0.8729922484206698, -3.104470338566296)
195 hurry (0.26905343515273183, -0.8446625769655788)
196 feedback (1.4770959357012081, 1.1574892770115728)
197 provide (0.05043495604275932, 1.138817896937156)
198 my (0.5154910800760721, -0.61638648838802)
199 waiting (-1.1430251381684502, -0.719808415687556)
200 singles (-0.5917418563965158, -1.4861190620054108)
201 hot (-0.26543862973628723, 3.8643817477017666)
202 2 (0.3613957149983489, 0.8721055746515962)
203 rescheduled (-2.4203520411134467, -1.5635289728788784)
204 approved (0.19334923268007817, -0.3720965561005001)
205 application (1.9427338339882576, -1.0804098728226947)
206 loan (-0.025413123972448406, 1.4450111810232564)
207 found (1.192489016201973, 1.5493647662345589)
208 money (-0.3834418213019612, 1.351681400644946)
209 trial (-1.5524165589577752, -0.11778302887209621)
210 interesting (-1.1359018966995098, 0.10514917119487033)
211 weight (-1.3465793025330386, 3.3336152414987166)
212 credit (-0.16444323908708847, -0.2811710296132253)
213 getaway (2.3908901454799234, 1.6537005551578712)
214 hours (1.2917359522861243, 1.9004845427310342)
215 24 (-0.46467998537783467, 0.49358295553529163)
216 suspended (2.441285416424999, -1.090448111627657)
217 will (0.4274896279046708, 1.7727165790548542)
218 act (-0.28156391854172136, 0.4016942474153012)
219 bill (0.4027852052716329, -0.8959417002498973)
220 electricity (0.1200098152509256, -0.09310505968469963)
221 pay (-1.4998227820494274, -3.6514642100418966)
222 friendly (0.27024586882762347, 0.03862419799696459)
223 online (2.0062113198864613, 0.12726987941541765)

224 surveys (-2.215205481137908, 0.6861806638502537)
225 taking (1.1141083494045134, 0.446857104373827)
226 cash (-1.4898490243288132, 0.3718985241882063)
227 extra (-1.2614478860680638, -1.9039336031434027)
228 delivery (-1.9640782440451445, -1.0659586786814161)
229 deals (0.06271022440050401, 1.2000486208622803)
230 phone (0.9353889536218302, 2.4782025019937937)
231 upgrade (3.5713697950630614, 0.7872095863997199)
232 ahead (-1.7513285522255346, -0.36109416363948654)
233 fantastic (2.432352906530467, -2.253001401949391)
234 wishing (-2.3868872110511035, 0.4082119198521477)
235 birthday (0.5462749108012288, -0.35153812703894277)
236 happy (-2.3756118315191594, -0.7016919561338856)
237 program (0.7242946594521441, 2.4445553709550767)
238 loss (0.2072365726263407, -0.3603938185788957)
239 article (0.9354376426982584, 3.1281272091354815)
240 check (-0.3316676869649907, -1.9303220611920864)
241 selected (1.1094309551169852, -1.0999195321541522)
242 voucher (1.7996837769402108, 4.645401711291978)
243 200 (-1.7702952765293454, 0.7644698218039133)
244 received (0.619412421210853, -2.7260524518366585)
245 secret (-1.0080621014496551, -1.5876514010965683)
246 ll (-0.14452999218699322, -1.187661052171448)
247 email (1.1460729202986661, -0.39920788480956354)
248 alert (-1.3900714961764975, 1.8493105330854445)
249 security (1.0376881756404672, -1.9664880888250384)
250 drill (-1.1334698887160666, -0.8343787776954864)
251 not (1.4034095564457276, 0.5128769489007309)
252 airport (-1.4153003868549294, 2.3560451810709804)
253 picnic (-0.4316910458678479, -0.747366308383802)
254 weather (2.370028341569733, -1.060609792836732)
255 immediately (-1.9429497078987785, -1.4004155211649332)
256 verify (-1.7990461416239367, -0.8681468102094226)
257 hold (-1.9016038096922971, 1.9728129219350117)
258 paypal (-0.7926501687603837, -1.5245795005029552)
259 slides (1.103484648362766, -2.3076799722688643)
260 presentation (-1.0959898339939993, -1.0737910623790985)
261 me (-2.6809428934565966, 0.4596126584749616)
262 send (0.3204899534858162, -1.4344962998145405)
263 two (-0.5061858852805475, -1.1773591235825)
264 promotion (-1.6951823934752273, -1.0538540026950916)
265 special (-2.2255051421835397, -0.8998048439465895)
266 charged (-0.6009475506200556, 2.263155986623412)
267 deadline (-0.619109303980688, -0.05930232966811777)
268 project (2.885340495871999, 2.7792660221558916)
269 ipad (0.4261650936203047, -1.114745475514181)
270 shopping (3.5700156083982946, -1.2099918190445995)
271 lunch (0.3063198892913068, 1.4059932646718265)
272 access (-0.6201224796602162, 1.1967057482244676)
273 1 (-1.2221261063098423, -1.5682556013289644)
274 streaming (-1.1255570343176167, -0.002586879758568979)
275 unlimited (2.094343780220457, 1.4039664222635921)
276 unlock (2.1348729670777367, 0.5282430381466283)
277 8 (-0.03193231574710788, -2.4646587194521627)
278 reservation (-0.4474416372153185, -1.1762495480570794)
279 monthly (-1.3134131787519596, -1.132648116429583)

```

280 000 (2.494172699820752, 0.586649260292753)
281 work (-0.5988482239023522, -0.24105589534225114)
282 later (0.3011279623372505, -1.9270383185082178)
283 collect (0.47412817706900884, -1.1064392488314214)
284 winnings (0.27737397960663496, -2.8449333171263826)
285 lottery (0.72331832662619, -1.1369575647828614)
286 unclaimed (0.058697772809444826, 2.2011310388267296)
287 friday (0.5928111114437131, 1.4477727810633978)
288 assignment (0.7864706134322947, -0.9362080965964553)
289 submit (0.464294932438481, -1.0067917165871245)
290 expiring (-0.7106757785014075, 2.241055203087936)
291 warranty (0.2264672609377217, -0.7812826654929068)
292 car (1.7037207637031724, 1.4233514187276777)
293 notice (2.5450922187164124, -2.3800727429732653)
294 final (0.5894705324872174, -0.3819879814899213)
295 refund (1.5029370667809625, 3.152320379989081)
296 talk (0.6558275224926774, -0.17246066350889078)
297 well (-0.325936410018356, -1.5797302160102062)
298 90 (-0.47389566279214457, 2.1294077421872757)
299 starts (-1.949412702900946, -0.7484835930499912)
300 mega (-1.491106994995411, -0.09720081411948917)
301 was (-1.5726004184571294, 0.535257641069902)

```

```

In [243... import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from gensim.models import Word2Vec

```

```

In [ ]: import numpy as np

def get_sentence_embedding(tokens):
    embeddings = []
    for token in tokens:
        if token in word_vectors:
            embeddings.append(word_vectors[token])
    if embeddings:
        return np.mean(embeddings, axis=0)
    else:
        return np.zeros(word2vec_model.vector_size)

df['embeddings'] = df['tokens'].apply(lambda x: get_sentence_embedding(x))

```

```

In [247... X = np.array(df['embeddings'].tolist())
y = df['category'].apply(lambda x: 1 if x == 'spam' else 0)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

```

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_scaled, y_train)

y_pred = knn.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy of KNN model: {accuracy * 100:.2f}%')

def predict_message(message):

    tokens = message.split()
    embeds = get_sentence_embedding(tokens)
    embeds_scaled = scaler.transform([embeds])
    prediction = knn.predict(embeds_scaled)
    return 'spam' if prediction[0] == 1 else 'ham'

print(predict_message("Congratulations! You won a free iPhone!"))
print(predict_message("You have arrived at the destination"))
```

Accuracy of KNN model: 73.33%

spam

ham

In []:

In []:

In []:

In []:

Create a basic nlp program to find the words, phrases, names and concepts using spaCy. Create an english nlp object.
Process the text and initialize a doc object in the variable name. Select the first token of the doc and print its text.

```
pip install spacy
```

```

Requirement already satisfied: spacy in /usr/local/lib/python3.10/dist-packages (3.7.4)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.10/dist-packages (from spacy) (3.0.11)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.0.0)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.0.10)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.10/dist-packages (from spacy) (2.0.7)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.10/dist-packages (from spacy) (3.0.3)
Requirement already satisfied: thinc<8.3.0,>=8.2.2 in /usr/local/lib/python3.10/dist-packages (from spacy) (8.2.2)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.0.0)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.10/dist-packages (from spacy) (2.4.3)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.10/dist-packages (from spacy) (2.0.6)
Requirement already satisfied: weasel<0.4.0,>=0.1.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (0.4.0)
Requirement already satisfied: typer<0.10.0,>=0.3.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (0.10.0)
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in /usr/local/lib/python3.10/dist-packages (from spacy) (7.0.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (4.66.1)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (2.32.0)
Requirement already satisfied: pydantic!=1.8,!1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.10/dist-packages (from spacy) (2.10.6)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from spacy) (3.1.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from spacy) (67.7.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (23.2)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (3.3.0)
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.25.2)
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from pydantic) (0.6.0)
Requirement already satisfied: pydantic-core==2.16.3 in /usr/local/lib/python3.10/dist-packages (from pydantic) (2.16.3)
Requirement already satisfied: typing-extensions>=4.6.1 in /usr/local/lib/python3.10/dist-packages (from pydantic) (4.11.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests) (3.3.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests) (3.10.1)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests) (2025.1.1)
Requirement already satisfied: blis<0.8.0,>=0.7.8 in /usr/local/lib/python3.10/dist-packages (from thinc<8.3.0,>=8.2.2) (0.7.10)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.10/dist-packages (from thinc<8.3.0,>=8.2.2) (0.0.4)
Requirement already satisfied: click<9.0.0,>=7.1.1 in /usr/local/lib/python3.10/dist-packages (from typer<0.10.0,>=0.3.0) (8.1.7)
Requirement already satisfied: cloudpathlib<0.17.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from weasel<0.4.0,>=0.1.0) (0.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->spacy) (2.1.5)

```

```
!python -m spacy download en_core_web_sm
```

```

Collecting en-core-web-sm==3.7.1
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.7.1/en_core_web_sm-3.7.1.tar.gz:
    12.8/12.8 MB 28.4 MB/s eta 0:00:00
Requirement already satisfied: spacy<3.8.0,>=3.7.2 in /usr/local/lib/python3.10/dist-packages (from en-core-web-sm==3.7.1) (3.7.4)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2) (3.0.11)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2) (1.0.0)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2) (1.0.10)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2) (2.0.7)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2) (3.0.3)
Requirement already satisfied: thinc<8.3.0,>=8.2.2 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2) (8.2.2)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2) (1.0.0)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2) (2.4.3)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2) (2.0.6)
Requirement already satisfied: weasel<0.4.0,>=0.1.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2) (0.4.0)
Requirement already satisfied: typer<0.10.0,>=0.3.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2) (0.10.0)
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2) (7.0.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2) (4.66.1)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2) (2.32.0)
Requirement already satisfied: pydantic!=1.8,!1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2) (2.10.6)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2) (3.1.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2) (67.7.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2) (23.2)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2) (3.3.0)
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2) (1.25.2)
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from pydantic) (0.6.0)
Requirement already satisfied: pydantic-core==2.16.3 in /usr/local/lib/python3.10/dist-packages (from pydantic) (2.16.3)
Requirement already satisfied: typing-extensions>=4.6.1 in /usr/local/lib/python3.10/dist-packages (from pydantic) (4.11.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests) (3.3.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests) (3.10.1)

```

Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3
 Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3
 Requirement already satisfied: blis<0.8.0,>=0.7.8 in /usr/local/lib/python3.10/dist-packages (from thinc<8.3.6
 Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.10/dist-packages (from thinc<8.3.6
 Requirement already satisfied: click<9.0.0,>=7.1.1 in /usr/local/lib/python3.10/dist-packages (from typer<0.16
 Requirement already satisfied: cloudpathlib<0.17.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from we
 Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->spacy)

✓ Download and installation successful

You can now load the package via spacy.load('en_core_web_sm')

⚠ Restart to reload dependencies

If you are in a Jupyter or Colab notebook, you may need to restart Python in order to load all the package's dependencies. You can do this by selecting the 'Restart kernel' or 'Restart runtime' option.

```
import spacy
```

```
# Load English language model
nlp = spacy.load("en_core_web_sm")
```

```
# Process the text
```

```
text = "Amidst the bustling city streets, where cars honk incessantly and pedestrians hurry along the sidewalks, I  

doc = nlp(text)
```

```
# Extract words
```

```
words = [token.text for token in doc if not token.is_punct]
print("Words:", words)
```

```
⇒ Words: ['Amidst', 'the', 'bustling', 'city', 'streets', 'where', 'cars', 'honk', 'incessantly', 'and', 'pedest
```

```
# Extract sentences
```

```
sentences = [sent.text for sent in doc.sents]
```

```
print("Sentences:", sentences)
```

```
⇒ Sentences: ['Amidst the bustling city streets, where cars honk incessantly and pedestrians hurry along the sid
```

```
# Extract parts of speech
```

```
for token in doc:
    print(f"TOKEN:{token.text},POS:{token.pos_}",end=" ")
```

```
⇒ TOKEN:Amidst,POS:ADP TOKEN:the,POS:DET TOKEN:bustling,POS:ADJ TOKEN:city,POS:NOUN TOKEN:streets,POS:NOUN
```

```
# stop words removal
```

```
stop_word=[token.text for token in doc if not token.is_stop]
stop_words=' '.join(stop_word)
print(f"original text: {text}")
print(f"Filtered text: {stop_words}")
```

```
⇒ original text: Amidst the bustling city streets, where cars honk incessantly and pedestrians hurry along the s  

  Filtered text: Amidst bustling city streets , cars honk incessantly pedestrians hurry sidewalks , lies serene
```

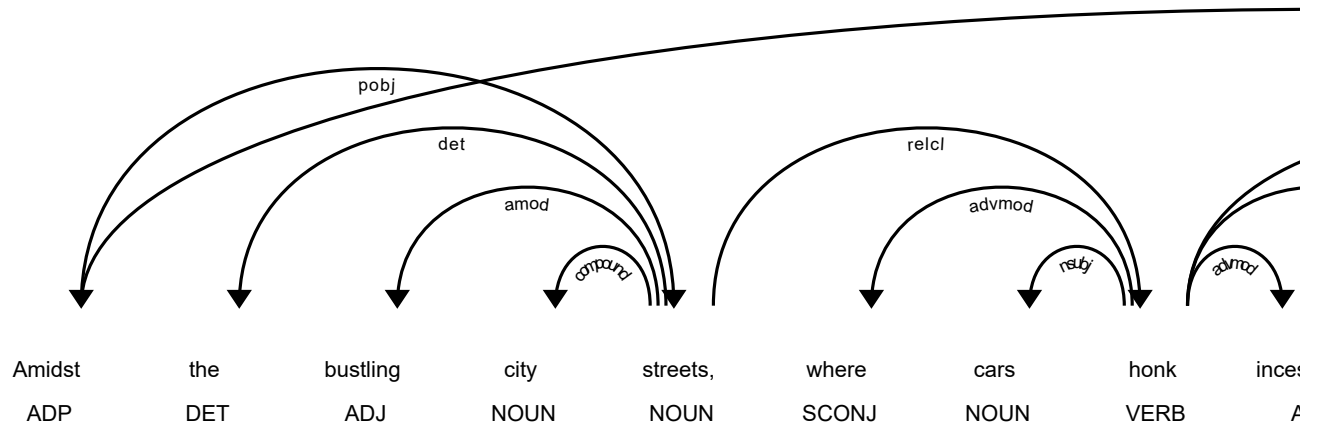
```
# count the stop words
```

```
stop_words_before = sum(1 for token in doc if token.is_stop)
doc_f=nlp(stop_words)
stop_words_after = sum(1 for token in doc_f if token.is_stop)
print("stop_words_before",stop_words_before)
print("stop_words_after ",stop_words_after)
```

```
⇒ stop_words_before 70  

  stop_words_after 0
```

```
# parsing dependency in visual format
from spacy import displacy
displacy.render(doc, style="dep", options={"distance":100})
```



Program5: Perform basic frequency analysis on a text

- i. Count the frequency of each word in a given text.
- ii. Visualize the word frequencies using a bar chart.

What is Frequency Analysis?

Frequency analysis in Natural Language Processing (NLP) is a fundamental technique to extract insights from text data by identifying the most commonly occurring words.

Why is it useful?

Helps in understanding dominant themes in a text.

Useful for keyword extraction and text summarization.

Forms the basis for more advanced NLP tasks like text classification, sentiment analysis, and topic modeling.

How is it done?

Tokenization: Splitting text into individual words.

Normalization: Lowercasing and removing punctuation/special characters.

Counting Words: Using frequency counts (e.g., Counter from Python's collections module).

Visualization: Using graphs like bar charts or word clouds to represent common words visually.

Step 1: Preprocessing the Text

Before performing frequency analysis, text data needs to be cleaned and processed.

Convert text to lowercase (to avoid counting "NLP" and "nlp" separately).

Remove special characters and punctuation (to count words properly). Tokenize the text (split it into individual words).

Optionally, remove stopwords (common words like "the," "is," "and" that do not add much meaning).

Step 2: Counting Word Frequencies

After preprocessing, we count how many times each word appears.

The Counter() class from the collections module helps efficiently count word occurrences.

We can extract the top N most common words to focus on the most relevant words.

Step 3: Visualizing the Data

Why visualize word frequency? Helps in identifying trends and important words at a glance. Useful in market research, and social media analysis.

Common visualization techniques:

Bar charts: Best for structured data representation.

Word clouds: Show the importance of words based on size.

Histograms: Display the distribution of word frequencies.

```
import re
import matplotlib.pyplot as plt
from collections import Counter

# Sample text
text = """Natural Language Processing (NLP) is a field of artificial intelligence.
NLP allows computers to understand human language. NLP techniques include tokenization,
part-of-speech tagging, named entity recognition, and machine translation."""

# Preprocessing: Convert text to lowercase and remove special characters
text = text.lower()
text = re.sub(r'[^a-z\s]', '', text) # Remove punctuation

# Tokenize words
words = text.split()

# Count word frequencies
word_counts = Counter(words)

# Display top 10 most common words
print("Top 10 words:", word_counts.most_common(10))

# Visualization using Bar Chart
plt.figure(figsize=(10, 5))
```



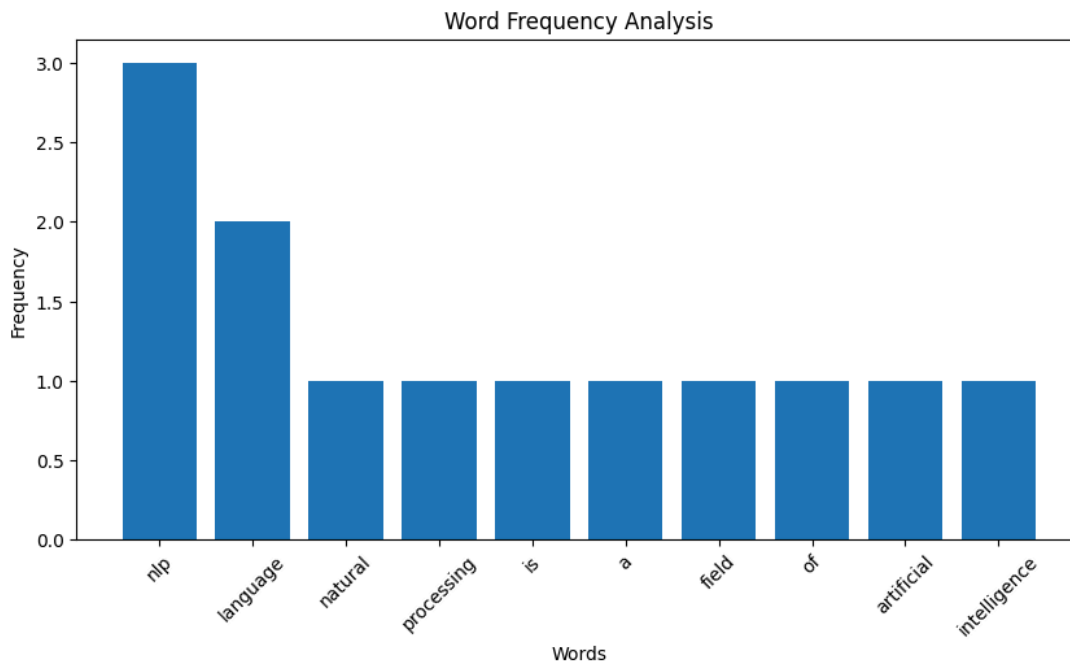
McAfee | WebAdvisor



Your download's being scanned.
We'll let you know if there's an issue.

```
plt.bar(*zip(*word_counts.most_common(10))) # Plot top 10 words
plt.xlabel("Words")
plt.ylabel("Frequency")
plt.title("Word Frequency Analysis")
plt.xticks(rotation=45)
plt.show()
```

Top 10 words: [('nlp', 3), ('language', 2), ('natural', 1), ('processing', 1), ('is', 1), ('a', 1), ('field', 1), ('of', 1), ('artificial', 1), ('intelligence', 1)]



```
import re
import nltk
import matplotlib.pyplot as plt
from collections import Counter
from nltk.corpus import stopwords
from wordcloud import WordCloud

# Download stopwords if not already available
nltk.download('stopwords')

# Define stopwords
stop_words = set(stopwords.words('english'))

# Sample text
text = """Natural Language Processing (NLP) is a field of artificial intelligence.
NLP allows computers to understand human language. NLP techniques include tokenization,
part-of-speech tagging, named entity recognition, and machine translation."""

# Preprocessing: Convert to lowercase and remove special characters
text = text.lower()
text = re.sub(r'^a-z\s', '', text) # Remove punctuation

# Tokenization and Stopword Removal
words = text.split()
filtered_words = [word for word in words if word not in stop_words] # Remove stopwords

# Count word frequencies
word_counts = Counter(filtered_words)

# Display top 10 most common words
print("Top 10 words:", word_counts.most_common(10))

# Bar Chart Visualization
plt.figure(figsize=(10, 5))
plt.bar(*zip(*word_counts.most_common(10))) # Plot top 10 words
plt.xlabel("Words")
plt.ylabel("Frequency")
plt.title("Word Frequency Analysis (After Stopword Removal)")
plt.xticks(rotation=45)
plt.show()

# Generate Word Cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate_from_freq

# Display Word Cloud
plt.figure(figsize=(10, 5))
```



McAfee | WebAdvisor

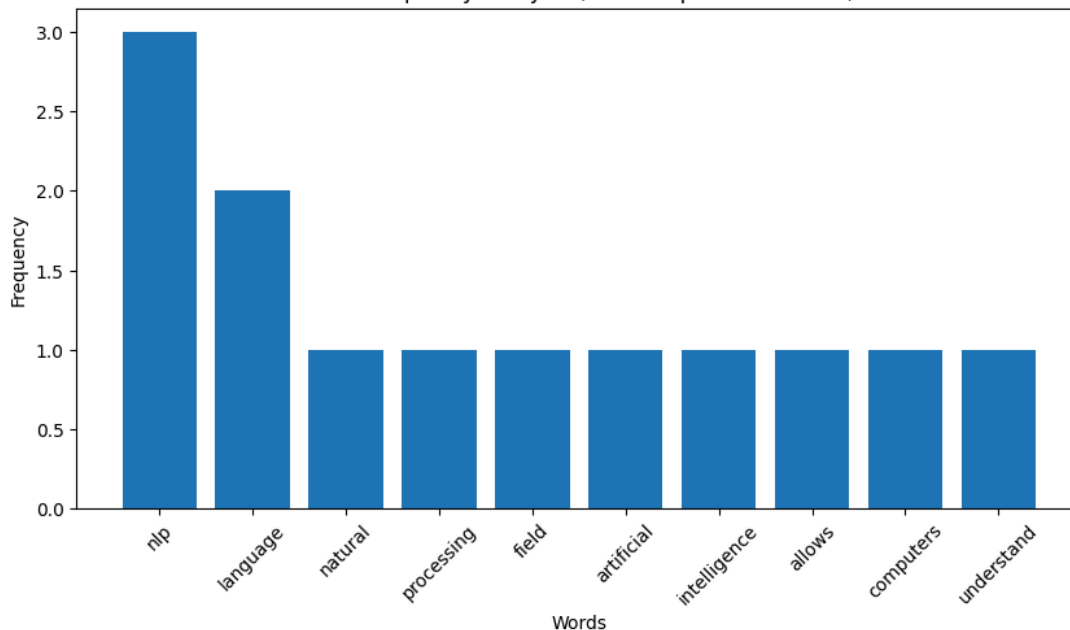


Your download's being scanned.
We'll let you know if there's an issue.

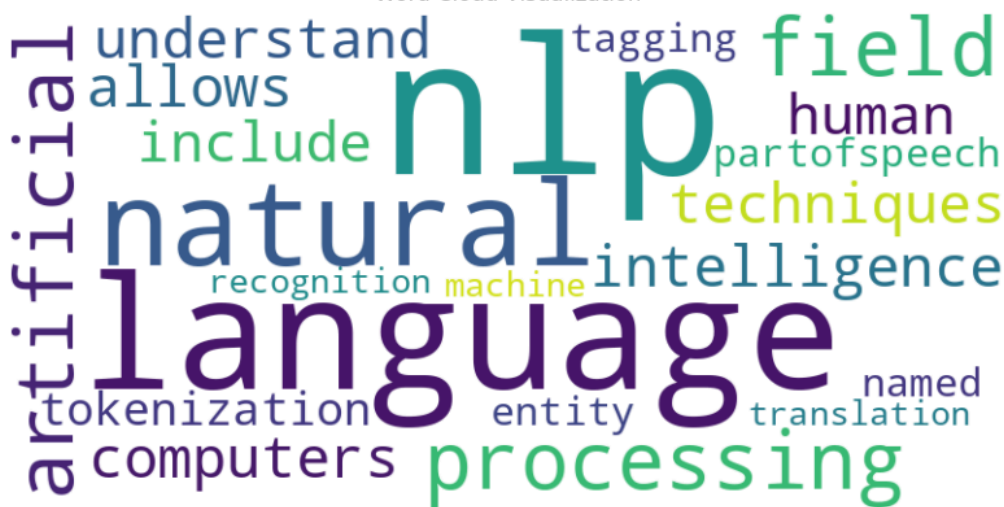

```
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title("Word Cloud Visualization")
plt.show()
```

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.
Top 10 words: [('nlp', 3), ('language', 2), ('natural', 1), ('processing', 1), ('field', 1), ('artificial', 1), ('intelligence', 1), ('allows', 1), ('computers', 1), ('understand', 1)]

Word Frequency Analysis (After Stopword Removal)



Word Cloud Visualization



McAfee | WebAdvisor



Your download's being scanned.
We'll let you know if there's an issue.

Program4: Measure similarity between texts. Use vectorization techniques (e.g., TF-IDF, word embeddings). i. Implement cosine similarity ii. Compare the effectiveness of different similarity measures

Objective:

To measure the similarity between texts using different vectorization techniques (TF-IDF and word embeddings) and implement cosine similarity to compare the effectiveness of different similarity measures.

Approach:

Text Vectorization Techniques: TF-IDF (Term Frequency-Inverse Document Frequency): Converts text into numerical features based on term importance.

Word Embeddings (Word2Vec, GloVe, or BERT): Captures semantic meaning by mapping words into high-dimensional vector space.

Similarity Measures:

Cosine Similarity: Measures the cosine of the angle between two vectors (ranges from 0 to 1, where 1 means identical).

Euclidean Distance: Measures the absolute difference between vector points in space.

Jaccard Similarity: Measures the intersection over the union of word sets.

TF-IDF is a numerical statistic used to evaluate the importance of a word in a document relative to a collection of documents (corpus). It is widely used in information retrieval, search engines, and text similarity tasks to represent text as numerical vectors.

TF-IDF consists of two main parts:

a) Term Frequency (TF):

TF measures how frequently a word appears in a document. The assumption is that words appearing frequently in a document are important.

B) Inverse Document Frequency (IDF):

IDF measures the importance of a word across all documents. Common words (e.g., "the", "is", "and") have high TF but are not informative, so IDF penalizes them.

$$TF(w) = \frac{\text{Number of times the word } w \text{ appears in the document}}{\text{Total number of words in the document}}$$

Suppose a document contains 100 words, and the word "machine" appears 5 times.

$$TF(\text{"machine"}) = \frac{5}{100} = 0.05$$

$$IDF(w) = \log \left(\frac{\text{Total number of documents}}{\text{Number of documents containing the word } w} \right)$$

If a word appears in all documents, IDF becomes low (close to 0), reducing its importance.



If a word appears in very few documents, IDF is high, increasing its weight. Example:

Suppose we have 10,000 documents, and "machine" appears in 100 documents

$$IDF(\text{"machine"}) = \log\left(\frac{10,000}{100}\right) = \log(100) = 2$$

TF-IDF Score Calculation:

$$TF - IDF(w) = TF(w) \times IDF(w)$$

Example:

If the TF of "machine" is 0.05 and its IDF is 2, then

$$TF - IDF(\text{"machine"}) = 0.05 \times 2 = 0.1$$

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# Sample texts
text1 = "Machine learning is a field of artificial intelligence."
text2 = "Deep learning is a branch of artificial intelligence and machine learning."

# Convert texts to TF-IDF vectors
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform([text1, text2])

# Compute cosine similarity
cosine_sim = cosine_similarity(tfidf_matrix[0], tfidf_matrix[1])

print(f"Cosine Similarity (TF-IDF): {cosine_sim[0][0]:.4f}")
```

🔗 Cosine Similarity (TF-IDF): 0.6416

TF-IDF transforms text into numerical vectors.

Cosine similarity is computed as:

$$\text{Cosine Similarity} = \frac{A \cdot B}{\|A\| \|B\|}$$

A higher value means the texts are more similar.

Advantages of TF-IDF

- ✅ Handles common vs. rare words well – Reduces the importance of words like "the" and "is".
- ✅ Simple and efficient – Works well for search engines and document classification.
- ✅ Effective for keyword-based similarity – Helps in document ranking, topic modeling, and text clustering.

Implementing Cosine Similarity using Word Embeddings (Word2Vec - Pretrained GloVe vectors)

```
import numpy as np
import gensim.downloader as api
from sklearn.metrics.pairwise import cosine_similarity
```



```
# Load pretrained GloVe model
glove_model = api.load("glove-wiki-gigaword-50")

def get_embedding(text, model):
    words = text.lower().split()
    word_vectors = [model[word] for word in words if word in model]


    if not word_vectors:
        return np.zeros(model.vector_size)

    return np.mean(word_vectors, axis=0)

# Compute embeddings
embedding1 = get_embedding(text1, glove_model)
embedding2 = get_embedding(text2, glove_model)

# Compute cosine similarity
cosine_sim = cosine_similarity([embedding1], [embedding2])

print(f"Cosine Similarity (Word Embeddings - GloVe): {cosine_sim[0][0]:.4f}")
```

 [=====] 100.0% 66.0/66.0MB downloaded
Cosine Similarity (Word Embeddings - GloVe): 0.9705

- Pretrained **GloVe** vectors capture word meanings.
- Each text is represented as an **average of word vectors**.
- Cosine similarity is computed on these dense representations.

TF-IDF works well for keyword-based similarity but lacks meaning.

Word Embeddings capture semantics better, making them more effective for NLP tasks like sentiment analysis, search, and recommendation systems.

When to Use TF-IDF?

✅ Best for traditional NLP tasks like:

Information retrieval (search engines, document ranking)

Keyword-based text similarity (plagiarism detection, news categorization)

Text classification with small datasets

When to Use Word Embeddings?

✅ Best for deep NLP applications like:

Chatbots and virtual assistants (Google Assistant, Siri)

Machine Translation (Google Translate)

Sentiment analysis (understanding emotions in reviews, social media)

Text summarization, question-answering systems





program 3. Download Wikipedia's page on open source and convert the text to its native forms. Try it with various stemming and lemmatization modules. Use Python's timer module to measure their performance.

Objective: This program aims to analyze the efficiency of stemming and lemmatization in NLP by:

Downloading the Wikipedia summary of "Open Source."

Applying text preprocessing techniques such as stemming and lemmatization to convert words to their root forms.

Comparing performance by measuring execution time using Python's time module.

Methodology

Fetch Wikipedia Text: Retrieve the "Open Source" summary using the Wikipedia API.

Tokenize Text: Use spaCy for word segmentation.

Apply Stemming : Use PorterStemmer from nltk to stem words.

Apply Lemmatization: Use spaCy to lemmatize words.

Measure Execution Time: Use time module to compare performance of both techniques.

Display Results: Print sample outputs and execution times.

import libraries

```
!pip install wikipedia-api
```

```
Requirement already satisfied: wikipedia-api in /usr/local/lib/python3.11/dist-packages (0.8.1)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from wikipedia-api) (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->wikipedia-api) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->wikipedia-api) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->wikipedia-api) (2.3.0)
Requirement already satisfied: certifi<=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->wikipedia-api) (2025.1.31)
```

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Package punkt is already up-to-date!
True
```

```
nltk.download()
```

```
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package subjectivity to /root/nltk_data...
[nltk_data] Package subjectivity is already up-to-date!
[nltk_data] Downloading package swadesh to /root/nltk_data...
[nltk_data] Package swadesh is already up-to-date!
[nltk_data] Downloading package switchboard to /root/nltk_data...
[nltk_data] Package switchboard is already up-to-date!
[nltk_data] Downloading package tagsets to /root/nltk_data...
[nltk_data] Package tagsets is already up-to-date!
[nltk_data] Downloading package tagsets_json to /root/nltk_data...
[nltk_data] Package tagsets_json is already up-to-date!
[nltk_data] Downloading package timit to /root/nltk_data...
[nltk_data] Package timit is already up-to-date!
[nltk_data] Downloading package toolbox to /root/nltk_data...
[nltk_data] Package toolbox is already up-to-date!
[nltk_data] Downloading package treebank to /root/nltk_data...
[nltk_data] Package treebank is already up-to-date!
[nltk_data] Downloading package twitter_samples to /root/nltk_data...
[nltk_data] Package twitter_samples is already up-to-date!
[nltk_data] Downloading package udhr to /root/nltk_data...
[nltk_data] Package udhr is already up-to-date!
[nltk_data] Downloading package udhr2 to /root/nltk_data...
[nltk_data] Package udhr2 is already up-to-date!
[nltk_data] Downloading package unicode_samples to /root/nltk_data...
[nltk_data] Package unicode_samples is already up-to-date!
[nltk_data] Downloading package universal_tagset to /root/nltk_data...
[nltk_data] Package universal_tagset is already up-to-date!
[nltk_data] Downloading package universal_treebanks_v20 to
/root/nltk_data...
[nltk_data] Package universal_treebanks_v20 is already up-to-date!
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data] Package vader_lexicon is already up-to-date!
[nltk_data] Downloading package verbnet to /root/nltk_data...
[nltk_data] Package verbnet is already up-to-date!
[nltk_data] Downloading package verbnet3 to /root/nltk_data...
[nltk_data] Package verbnet3 is already up-to-date!
[nltk_data] Downloading package webtext to /root/nltk_data...
[nltk_data] Package webtext is already up-to-date!
[nltk_data] Downloading package wmt15_eval to /root/nltk_data...
[nltk_data] Package wmt15_eval is already up-to-date!
[nltk_data] Downloading package word2vec_sample to /root/nltk_data...
[nltk_data] Package word2vec_sample is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Package wordnet is already up-to-date!
[nltk_data] Downloading package wordnet2021 to /root/nltk_data...
[nltk_data] Package wordnet2021 is already up-to-date!
[nltk_data] Downloading package wordnet2022 to /root/nltk_data...
[nltk_data] Package wordnet2022 is already up-to-date!
[nltk_data] Downloading package wordnet31 to /root/nltk_data...
[nltk_data] Package wordnet31 is already up-to-date!
[nltk_data] Downloading package wordnet_ic to /root/nltk_data...
[nltk_data] Package wordnet_ic is already up-to-date!
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data] Package words is already up-to-date!
[nltk_data] Downloading package ycoe to /root/nltk_data...
[nltk_data] Package ycoe is already up-to-date!
Done downloading collection all
```

```
import time # Timer module for performance measurement
```

```
import wikipediaapi # Wikipedia API to fetch text
import spacy # NLP library for lemmatization
from nltk.stem import PorterStemmer # Stemming module from nltk
from nltk.tokenize import word_tokenize # Tokenization module
```

1. Fetch Wikipedia Content

```
# Step 1: Fetch Wikipedia content
def get_wikipedia_text(page_title):
    wiki_wiki = wikipediaapi.Wikipedia(user_agent="MyNLPPProject/1.0", language="en")
    page = wiki_wiki.page(page_title)
    return page.summary if page.exists() else ""

text = get_wikipedia_text("Keshav_Memorial_Institute_of_Technology")
```

This function retrieves the summary of the Wikipedia page "Open Source" using wikipediaapi.

If the page exists, it returns the summary text.

2. Tokenization

```
# Tokenize the text
tokens = word_tokenize(text)
```

Splits the text into individual words (tokens), making it easier to process.

3. Apply Stemming

```
# Step 2: Apply Stemming
stemmer = PorterStemmer()

start_stem = time.time() # Start timer
stemmed_words = [stemmer.stem(word) for word in tokens]
end_stem = time.time() # End timer
```

Uses PorterStemmer from nltk to convert words into their stemmed form.

Example: "running" → "run", "better" → "bet".

4. Apply Lemmatization

```
# Step 3: Apply Lemmatization
nlp = spacy.load("en_core_web_sm")

start_lem = time.time() # Start timer
doc = nlp(" ".join(tokens))
lemmatized_words = [token.lemma_ for token in doc]
end_lem = time.time() # End timer
```

Uses spaCy to perform lemmatization, which provides proper root words.

Example: "running" → "run", "better" → "good".

The time module calculates execution times for stemming and lemmatization.

Used to compare performance differences between the two techniques.

5. Display Results and Performance Comparison

```
# Step 4: Display Results
print("Original Text Sample:", tokens[:10])
print("Stemmed Words:", stemmed_words[:10])
print("Lemmatized Words:", lemmatized_words[:10])

# Step 5: Performance Comparison
print("\nPerformance Analysis:")
print(f"Stemming Execution Time: {end_stem - start_stem:.5f} seconds")
print(f"Lemmatization Execution Time: {end_lem - start_lem:.5f} seconds")
```



```
Original Text Sample: ['Keshav', 'Memorial', 'Institute', 'of', 'Technology', 'is', 'a', 'private', 'engineering', 'college']
Stemmed Words: ['keshav', 'memori', 'institut', 'of', 'technolog', 'is', 'a', 'privat', 'engin', 'colleg']
Lemmatized Words: ['Keshav', 'Memorial', 'Institute', 'of', 'Technology', 'be', 'a', 'private', 'engineering', 'college']
```

```
Performance Analysis:
Stemming Execution Time: 0.00114 seconds
Lemmatization Execution Time: 0.02289 seconds
```


program 2. Using Python libraries, download Wikipedia's page on open source and tokenize the text, remove the stop words. What percentage of the page is stop words?

Objective:

The program aims to analyze the proportion of stop words in a Wikipedia article by:

Fetching the Wikipedia summary of "Open Source" using the Wikipedia API.

Tokenizing the text into individual words using the spaCy NLP library.

Identifying and removing stop words (commonly used words like "the," "is," and "and" that do not carry significant meaning).

Calculating the percentage of stop words in the text.

Methodology:

The Wikipedia content is retrieved programmatically.

spaCy is used to tokenize the text and filter out stop words.

The total number of words and stop words are counted.

Finally, the percentage of stop words in the text is computed.

1. Download Wikipedia Page Content

```
import requests # To fetch Wikipedia content

url = "https://en.wikipedia.org/api/rest_v1/page/summary/Open_source"
response = requests.get(url) # Sending HTTP GET request
data = response.json() # Converting response to JSON format
text = data["extract"] # Extracting the main content
```

We use the Wikipedia API to fetch the summary of the "Open Source" page.

The response is in JSON format, and we extract the main text from the "extract" key.

2. Load spaCy's English Model

```
nlp = spacy.load("en_core_web_sm")
```

We load the pre-trained small English model in spaCy, which includes tokenization and stop-word detection.

3. Process the Text

```
doc = nlp(text)
```

The Wikipedia text is processed using spaCy, and it returns a Doc object containing structured NLP tokens.

4. Tokenization and Stop-word Removal

```
tokens = [token.text for token in doc] # Tokenizing the text
stop_words = [token.text for token in doc if token.is_stop] # Extracting stop words
```

doc contains individual tokens (words and punctuation). We extract all tokens and separately extract only stop words using token.is_stop.

5. Calculate Percentage of Stop Words

```
total_tokens = len(tokens)
stop_word_count = len(stop_words)
percentage_stop_words = (stop_word_count / total_tokens) * 100
```

$$\left(\frac{\text{stop word count}}{\text{total token count}} \right) \times 100$$

We count the total words and stop words. The percentage of stop words is calculated as:

6. Display Results

```
print(f"Total words: {total_tokens}")
print(f"Stop words: {stop_word_count}")
print(f"Percentage of stop words: {percentage_stop_words:.2f}%")
```

```
Total words: 118
Stop words: 40
Percentage of stop words: 33.90%
```

The total words, stop words, and percentage of stop words are displayed.

```
import requests # To fetch Wikipedia content
import spacy # For NLP processing

# Step 1: Download Wikipedia page content
url = "https://en.wikipedia.org/api/rest_v1/page/summary/Open source"
```



```
response = requests.get(url) # Sending HTTP GET request
data = response.json() # Converting response to JSON format
text = data["extract"] # Extracting the main content


# Step 2: Load spaCy's English model
nlp = spacy.load("en_core_web_sm")

# Step 3: Process text using spaCy
doc = nlp(text)

# Step 4: Tokenization and Stop-word Removal
tokens = [token.text for token in doc] # Tokenizing the text
stop_words = [token.text for token in doc if token.is_stop] # Extracting stop words

# Step 5: Calculate Percentage of Stop Words
total_tokens = len(tokens)
stop_word_count = len(stop_words)
percentage_stop_words = (stop_word_count / total_tokens) * 100

# Display results
print(f"Total words: {total_tokens}")
print(f"Stop words: {stop_word_count}")
print(f"Percentage of stop words: {percentage_stop_words:.2f}%")
```

 Total words: 118
Stop words: 40
Percentage of stop words: 33.90%



program 1.Create a basic NLP program to find words, phrases, names and concepts using "spacy.blank" to create the English nlp object. Process the text and instantiate a Doc object in the variable doc. Select the first token of the Doc and print its text

Natural Language Processing (NLP) is a field of artificial intelligence that focuses on the interaction between computers and humans through natural language. spaCy is a popular Python library used for NLP tasks such as tokenization, part-of-speech tagging, named entity recognition, and more.

In this example, we will create a basic NLP program using spaCy to process a text, tokenize it, and extract the first token from the processed document.

Objective: The program demonstrates basic Natural Language Processing (NLP) using spaCy, focusing on:

Tokenization : Breaking text into individual words (tokens).

Phrase and Name Recognition: Identifying meaningful sequences of words.

Concept Extraction: Processing text for key terms.

Basic spaCy Functionality: Using spaCy.blank("en") to create a lightweight NLP pipeline.

Methodology

Initialize NLP Object: Create a blank English NLP pipeline using spacy.blank("en").

Process Input Text: Pass a sample text to the NLP object, generating a Doc object.

Tokenization: Extract individual words (tokens) from the text using doc.

Retrieve First Token: Access and print the first token using doc[0].text.

Output Generation: Display the extracted word, verifying successful text processing.

1. Import spaCy Library

```
import spacy
```

spaCy is a powerful NLP library used for text processing, tokenization, named entity recognition, and more.

2. Create a Blank English NLP Object:

```
nlp = spacy.blank("en")
```

spacy.blank("en") creates an empty English NLP pipeline. This means no pre-trained models or extra functionalities (like named entity recognition or POS tagging) are loaded. This is useful when you want to process raw text efficiently with minimal overhead.

3.Process the Text

```
text = "Asha is in love with Natural Language Processing."  
doc = nlp(text)
```

The text "John is learning Natural Language Processing." is passed to the NLP object.

doc = nlp(text) creates a Doc object that holds the processed text.

A Doc object is a sequence of Token objects, where each token represents a word or punctuation in the text.

4. Select and Print the First Token:

```
first_token = doc[0]
print("First token:", first_token.text)
```

↗ First token: Asha

`doc[0]` accesses the first token (word) in the processed text.

`first_token.text` extracts and prints the actual word.

```
# Import the spaCy library
import spacy

# Create an English NLP object using spacy.blank
nlp = spacy.blank("en")

# Define a sample text to process
text = "Asha is in love with Natural Language Processing."

# Process the text using the nlp object to create a Doc object
doc = nlp(text)

# The Doc object is a container for accessing linguistic annotations
# It contains tokens, which are individual words, punctuation marks, etc.

# Select the first token of the Doc object
first_token = doc[0]

# Print the text of the first token
print("First token:", first_token.text)
```

↗ First token: Asha