AI Systems - Using Machine Learning to find Patterns in Text.

Overview

Word2Vec is a natural language processing technique developed by Google in two papers published in 2013. It is made up of models for mapping words to vectors of real numbers, or for creating embeddings. Word embeddings work on the premise that words that appear in a similar context in a text tend to be near to each other in vector space. Using vector representations of words in machine learning models has since become a generally accepted standard.

The task here is to perform text analysis on the given dataset of *Coronavirus News Headlines* using Word2Vec model and PCA for data visualization. The goal is to find patterns and clusters in the text data that can help in understanding the topics and themes of the news articles. The choice of using Word2Vec and PCA is justified as Word2Vec is a widely used method for creating word embeddings and PCA is a useful technique for reducing the dimensionality of the data for visualization purposes.

Text Data

	title
0	British coronavirus evacuee who was 'taken ill' on last flight out of Wuhan
1	CDC files for emergency approval of its coronavirus test
2	Can ANY face mask really protect you from the coronavirus?
3	Experts: You won't catch coronavirus from packages from China
4	Seventh American diagnosed with coronavirus as US declares public health emergency
...	...
161649	Why have deaths stayed steady while infections are spiking?
161650	Trump says three novel coronavirus vaccine candidates looking really good
161651	Coronavirus: Trump touts response as COVID-19 daily cases surge
161652	Woman, 21, 'attacked five airline ticketing agents, coughed on cops while claiming to have COVID-19'
161653	Why Morehouse and Clemson Made Different Football Choices

The above dataset consists of a single column of news headlines related to Coronavirus. It contains 896 rows and was collected from various news sources. The data was **pre-processed** by *removing stopwords, punctuations, and special characters, converted to lowercase and then tokenized*. After which it appeared like below:

	title
0	british coronavirus evacuee taken ill last flight wuhan
1	cdc files emergency approval coronavirus test
2	face mask really protect coronavirus
3	experts catch coronavirus packages china
4	seventh american diagnosed coronavirus us declares public health emergency
...	...
161649	deaths stayed steady infections spiking
161650	trump says three novel coronavirus vaccine candidates looking really good
161651	coronavirus trump touts response covid daily cases surge
161652	woman attacked five airline ticketing agents coughed cops claiming covid
161653	morehouse clemson made different football choices

AI Systems - Using Machine Learning to find Patterns in Text.

Method

The Word2Vec model was trained on the pre-processed dataset using the *gensim* library in Python. Word2Vec is a neural network-based method for generating word embeddings. It learns the vector representation of words based on their co-occurrence in the text corpus. The ideal number of dimensions is determined by the **size** of the dataset. In our instance, 100 dimensions appear to be quite effective. The minimal frequency of words is controlled by the **count** parameter. All words with a total frequency less than this number are ignored during model training.

```
[['british', 'coronavirus', 'evacuee', "'taken", 'ill', 'last', 'flight', 'wuhan'], ['cdc', 'files', 'emergency', 'approval',
'coronavirus', 'test'], ['face', 'mask', 'really', 'protect', 'coronavirus'], ['experts', 'wo', "n't", 'catch', 'coronavirus',
'packages', 'china'], ['seventh', 'american', 'diagnosed', 'coronavirus', 'us', 'declares', 'public', 'health', 'emergency'],
['rep.', 'paul', 'gosar', 'coronavirus', 'save', 'lives', 'must', 'restrict', 'travel', 'affected', 'areas', 'us'], ['cambodia
n', 'pm', 'says', 'visit', 'students', 'china', "'s", 'wuhan', 'moral', 'support'], ['burdened', 'sanctions', 'north', 'korea',
'sees', 'coronavirus', 'threaten', 'economic', 'lifelines'], ['two', 'coronavirus', 'cases', 'confirmed', 'uk'], ['coronaviru
s', 'worldwide', 'cases', 'surpass', 'sars', 'outbreak', '2003'], ['coronavirus', 'catching', 'diseases', 'animals'], ['coronav
irus', 'cement', 'mixers', 'become', 'celebrities', 'china', 'lockdown'], ['coronavirus', 'wuhan', 'diary', 'living', 'alone',
'city', 'gone', 'quiet'], ['coronavirus', 'declared', 'global', 'health', 'emergency'], ['coronavirus', 'death', 'toll', 'rise
s', 'virus', 'spreads', 'every', 'chinese', 'region'], ['coronavirus', 'quarantine', 'mean'], ['coronavirus', 'scientists', 'ra
ce', 'develop', 'vaccine'], ['coronavirus', 'us', 'laboratory', 'developing', 'vaccine'], ['wuhan', 'london-sized', 'city', 'vi
rus', 'began'], ['china', 'coronavirus', 'lessons', 'learned', 'sars', 'outbreak']]
```

The **trained** Word2Vec model was then used to find the **closest n words** to the query word like in our case it is the word ‘economic’. Normally, Word2vec uses following two architectures to achieve this, first **CBOW or Continuous Bag of words** where the model predicts the word under consideration given context words within specific window and **Skip Gram** where the model predicts embeddings for the surrounding context words in the specific window given a current word. We have used Skip-Gram.

```
('fed', 0.8753296732902527),
('recession', 0.8284371495246887),
('fiscal', 0.8150244355201721),
('pain', 0.8105964660644531),
('economy', 0.8036598563194275),
('rebound', 0.7958559989929199),
('focus', 0.7948346138000488),
('trade', 0.7867645025253296),
('growth', 0.7866014838218689),
('financial', 0.7856122851371765)]
```

And finally, was **tested** to find the near similar positive words for ‘outbreak’:

```
model.wv.most_similar(positive=['outbreak'], topn = 7)
```

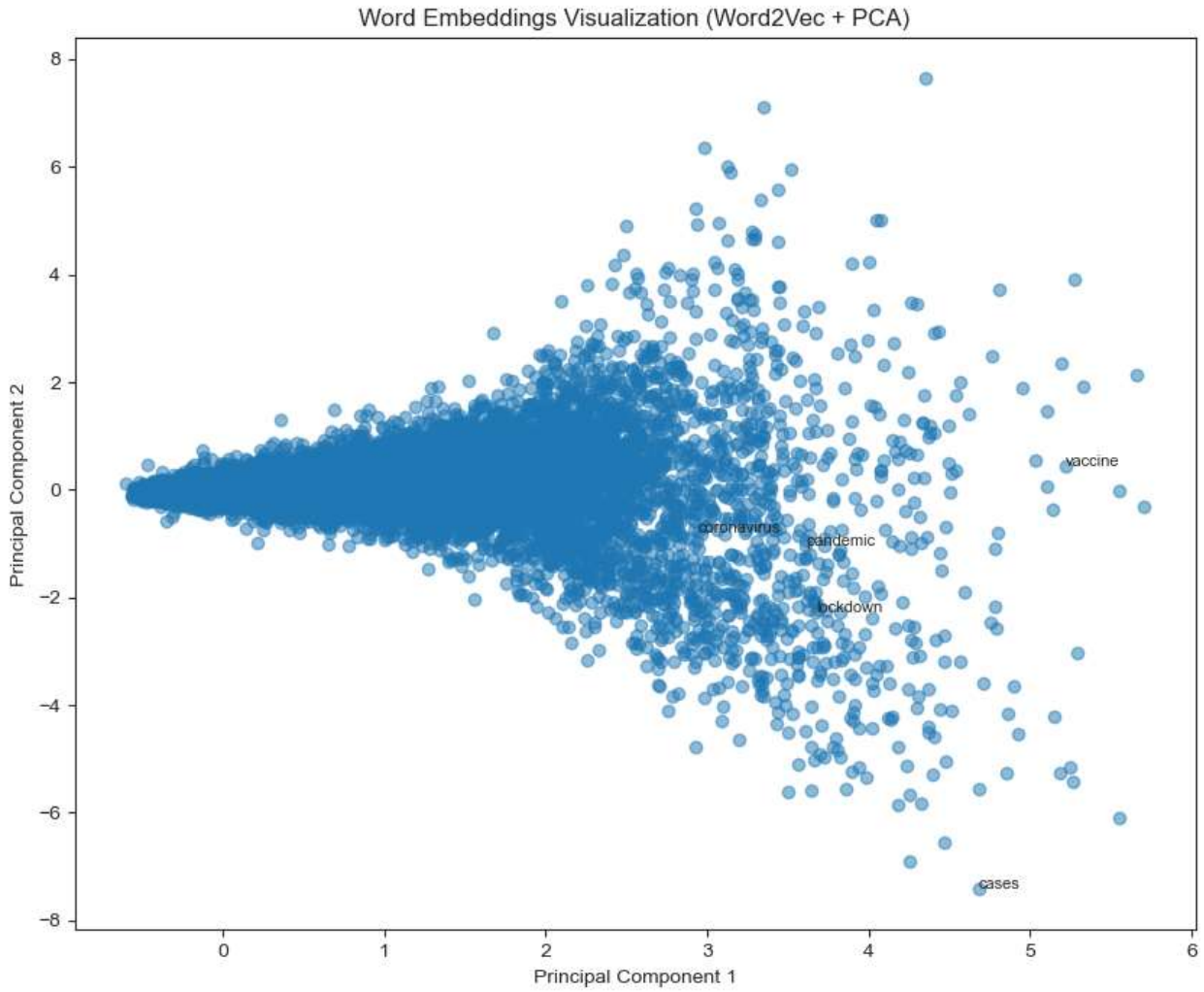
```
[('pandemic', 0.8574196100234985),
 ('epidemic', 0.7683645486831665),
 ('crisis', 0.7472087740898132),
 ("'distraction", 0.7444688677787781),
 ('concerns', 0.7219387292861938),
 ('chaos', 0.7175068855285645),
 ('scare', 0.7042276263237)]
```

AI Systems - Using Machine Learning to find Patterns in Text.

Evaluation and findings

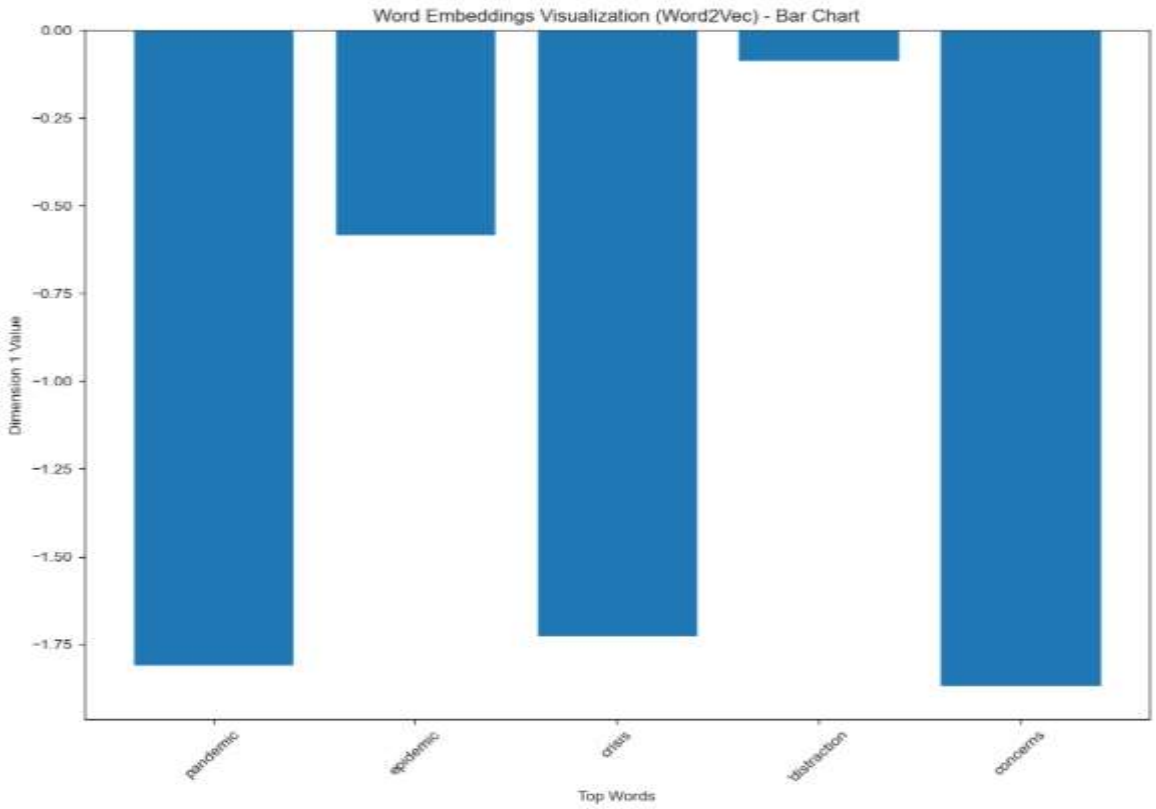
PCA or Principal Component Analysis has been used to **reduce the dimensionality** of the word embeddings generated by Word2Vec. PCA is a technique that extracts the important features from high-dimensional data and projects it onto a lower-dimensional space while preserving the variance in the data. It worked better than T-SNE, in our case, as PCA is a deterministic algorithm whereas T-SNE is non deterministic in nature (may produce different output each time). The reduced-dimensional data was then visualized using scatterplots and bar charts. The results of the analysis showed that the Word2Vec model was able to generate word embeddings that captured the semantic relationships between words in the corpus. The most similar words to the query word 'outbreak' included 'epidemic', 'pandemic', 'spread', 'cases' etc.

The scatterplots showed the distribution of the word embeddings in the reduced dimensional space, revealing distinct clusters of words.



Whereas the bar charts showed the frequency distribution of the most common words in the corpus, revealing the most common themes and topics of the news articles.

AI Systems - Using Machine Learning to find Patterns in Text.



In conclusion, the text analysis performed using Word2Vec and PCA on the Coronavirus news headlines dataset was able to identify distinct patterns and clusters in the text data. The Word2Vec model was able to generate word embeddings that captured the semantic relationships between words in the corpus, and the PCA technique was used to visualize the reduced-dimensional word embeddings. The analysis revealed distinct themes and topics related to Coronavirus, which can be useful for understanding the news coverage of the pandemic.

Python Program:

Embedding Model to study patterns in data for coronavirus-news headlines.

In [81]:

```
# import required libraries
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from gensim.models import Word2Vec
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import re
```

In [82]:

AI Systems - Using Machine Learning to find Patterns in Text.

```
# Download required NLTK resources
nltk.download('stopwords')
nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to C:\Users\Mishika Singh
[nltk_data]     Gaur\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to C:\Users\Mishika Singh
[nltk_data]     Gaur\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[82]:

```
True
```

In [83]:

```
# Load the dataset
df = pd.read_csv('corona_news.csv')

# Remove any missing values in the 'title' column
df.dropna(subset=['title'], inplace=True)

# Print and view the dataset before preprocessing
sentences=df['title']
print(sentences[:10])
```

```
0    British coronavirus evacuee who was 'taken ill...
1    CDC files for emergency approval of its corona...
2    Can ANY face mask really protect you from the ...
3    Experts: You won't catch coronavirus from pack...
4    Seventh American diagnosed with coronavirus as...
5    Rep. Paul Gosar: Coronavirus - To save lives, ...
6    Cambodian PM says he will visit students in Ch...
7    Burdened by sanctions, North Korea sees corona...
8                Two coronavirus cases confirmed in UK
9    Coronavirus: Worldwide cases surpass Sars outb...
Name: title, dtype: object
```

In [84]:

```
# DATA PRE-PROCESSING STAGE

# Convert text to lowercase and tokenize
df['title'] = df['title'].str.lower()
df['title'] = df['title'].apply(word_tokenize)

# Remove special characters
PUNCT_RE = re.compile(r'[^\w\s]+$')
def is_punct(string):
  return PUNCT_RE.match(string) is not None

df['title'] = df['title'].apply(lambda x: [word for word in x if not is_punct(word)])
```

```
# Remove stopwords
stop_words = set(stopwords.words('english'))
df['title'] = df['title'].apply(lambda x: [word for word in x if word not in stop_words])

# Print and view the dataset after preprocessing
sentences=df['title']
print(sentences[:10])
```

```
0    [british, coronavirus, evacuee, 'taken, ill, l...
1    [cdc, files, emergency, approval, coronavirus,...
2           [face, mask, really, protect, coronavirus]
3    [experts, wo, n't, catch, coronavirus, package...
4    [seventh, american, diagnosed, coronavirus, us...
5    [rep., paul, gosar, coronavirus, save, lives, ...
6    [cambodian, pm, says, visit, students, china, ...
7    [burdened, sanctions, north, korea, sees, coro...
8             [two, coronavirus, cases, confirmed, uk]
9    [coronavirus, worldwide, cases, surpass, sars,...
Name: title, dtype: object
```

In [85]:

```
# Create corpus out of the dataset
corpus = df['title'].tolist()

print(corpus[:10])
```

```
[['british', 'coronavirus', 'evacuee', "'taken", 'ill', 'last', 'flight', '
wuhan'], ['cdc', 'files', 'emergency', 'approval', 'coronavirus', 'test'],
['face', 'mask', 'really', 'protect', 'coronavirus'], ['experts', 'wo', "n'
t", 'catch', 'coronavirus', 'packages', 'china'], ['seventh', 'american', '
diagnosed', 'coronavirus', 'us', 'declares', 'public', 'health', 'emergency
'], ['rep.', 'paul', 'gosar', 'coronavirus', 'save', 'lives', 'must', 'rest
rict', 'travel', 'affected', 'areas', 'us'], ['cambodian', 'pm', 'says', 'v
isit', 'students', 'china', "'s", 'wuhan', 'moral', 'support'], ['burdened'
, 'sanctions', 'north', 'korea', 'sees', 'coronavirus', 'threaten', 'econom
ic', 'lifelines'], ['two', 'coronavirus', 'cases', 'confirmed', 'uk'], ['co
ronavirus', 'worldwide', 'cases', 'surpass', 'sars', 'outbreak', '2003']]
```

In [77]:

```
# Train Word2Vec model
# vector_size defines the number of dimensions of the embeddings and the default is 100.
# min_count defines the minimum count of words to consider when training the model; words with occurren
ce less than this count will be ignored. The default for min_count is 5.
# workers defines the number of partitions during training and the default workers is 3.
# sg defines the training algorithm, either CBOW(0) or skip gram(1). The default training algorithm is CBO
W.
model = Word2Vec(corpus, vector_size=100, min_count=1,  sg=1, workers=1)
model.wv.most_similar('economic')
```

Out[77]:

```
[('fed', 0.8543230891227722),
```

```
 ('recession', 0.8305264115333557),
 ('focus', 0.8166211843490601),
 ('downturn', 0.8118957877159119),
 ('fiscal', 0.8087626695632935),
 ('pain', 0.8076224327087402),
 ('unprecedented', 0.8036547899246216),
 ('economy', 0.7919361591339111),
 ('growth', 0.7882484793663025),
 ('imf', 0.7868074774742126)]
```

In [79]:

```
# Test Word2Vec model
model.wv.most_similar(positive=['outbreak'], topn = 7)
```

Out[79]:

```
[('pandemic', 0.8574196100234985),
 ('epidemic', 0.7683645486831665),
 ('crisis', 0.7472087740898132),
 ("'distraction", 0.7444688677787781),
 ('concerns', 0.7219387292861938),
 ('chaos', 0.7175068855285645),
 ('scare', 0.7042276263237)]
```

In [87]:

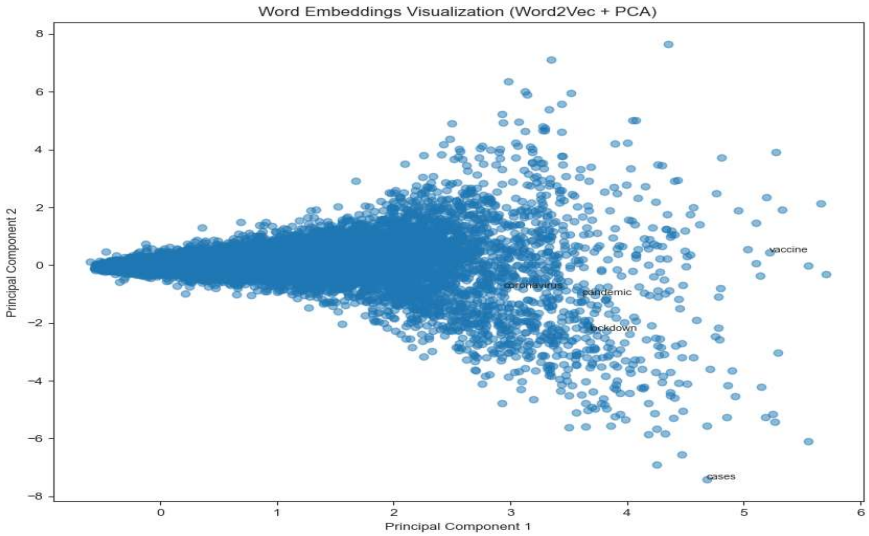
```
# Get word vectors and corresponding words
words = list(model.wv.key_to_index.keys())
print(words[:10])

X = model.wv[words]
```

```
['coronavirus', 'covid-19', "'s", 'says', 'new', 'trump', 'cases', 'lockdow
n', 'pandemic', 'china']
```

In [73]:

```
# Perform PCA on the word vectors
pca = PCA(n_components=2)
principal_components = pca.fit_transform(X)

# Create a new dataframe for visualization
df_pca = pd.DataFrame(principal_components, columns=['PC1', 'PC2'])
df_pca['word'] = words

# Visualization 1: Scatter Plot
plt.figure(figsize=(10, 8))
plt.scatter(df_pca['PC1'], df_pca['PC2'], alpha=0.5)

# Add labels to some data points for better interpretation
labels_to_show = ['coronavirus', 'pandemic', 'vaccine', 'lockdown', 'cases']
for label in labels_to_show:
```

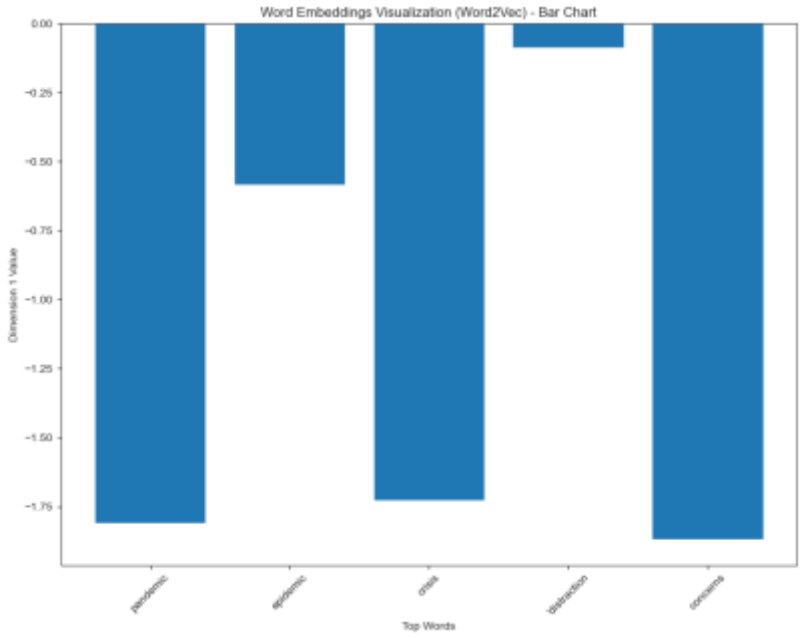
```
    x = df_pca.loc[df_pca['word'] == label, 'PC1'].values[0]
    y = df_pca.loc[df_pca['word'] == label, 'PC2'].values[0]
    plt.text(x, y, label, fontsize=8)

plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('Word Embeddings Visualization (Word2Vec + PCA)')
plt.show()
```



In [80]:

```
# Visualization 2: Bar Charts
plt.figure(figsize=(10, 8))
top_n = 5 # Number of top words to visualize
similar_words = model.wv.most_similar("outbreak")

# Extract and print only the words
words = [word for word, _ in similar_words]

top_words = words[:top_n]
x = range(top_n)
y = [model.wv[word][0] for word in top_words]  # Using the first dimension for visualization

plt.bar(x, y)
plt.xlabel('Top Words')
plt.ylabel('Dimension 1 Value')
plt.title('Word Embeddings Visualization (Word2Vec) - Bar Chart')
plt.xticks(x, top_words, rotation=45)
plt.tight_layout()
plt.show()
```

AI Systems - Using Machine Learning to find Patterns in Text.



s