Word Embedding Generation and Visualization using Word2Vec

1. Objective

To generate word embeddings from a given text corpus using the Word2Vec model and visualize the semantic relationships between words using PCA.

2. Technologies Used

- Python
- Gensim
- NLTK
- Scikit-learn
- Matplotlib

3. Theory

Word embeddings are dense vector representations of words that capture semantic and syntactic relationships. Unlike traditional one-hot encoding, embeddings place similar words closer in vector space. Word2Vec learns these representations using neural networks based on word co-occurrence in a corpus.

4. Types of Word Embeddings

- One-Hot Encoding
- TF-IDF
- Word2Vec (CBOW, Skip-gram)

- GloVe
- FastText

---

5. Algorithm / Workflow

1. Install required libraries
2. Prepare a text corpus
3. Tokenize text into words
4. Train Word2Vec model
5. Extract word vectors
6. Apply PCA for dimensionality reduction
7. Visualize embeddings in 2D space

---

6. Implementation (Code)

```
!pip install gensim scikit-learn matplotlib
from gensim.models import Word2Vec
from nltk.tokenize import word_tokenize
import nltk

nltk.download('punkt')
nltk.download('punkt_tab')

corpus = [
    "Machine learning enables computers to learn from data",
    "Natural language processing helps machines understand human
language",
    "Word embeddings represent words in continuous vector space",
    "Word2Vec learns semantic relationships between words",
    "NLP techniques are widely used in search engines and chatbots"
]

tokenized_corpus = [word_tokenize(sentence.lower()) for sentence in
corpus]

model = Word2Vec(sentences=tokenized_corpus, vector_size=100,
window=5, min_count=1)
model.save("word2vec.model")

model = Word2Vec.load("word2vec.model")

words = list(model.wv.index_to_key)[:10]
word_vectors = [model.wv[word] for word in words]

from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca_result = pca.fit_transform(word_vectors)

import matplotlib.pyplot as plt
plt.scatter(pca_result[:, 0], pca_result[:, 1])
for i, word in enumerate(words):
    plt.annotate(word, (pca_result[i, 0], pca_result[i, 1]))
plt.show()
```

---

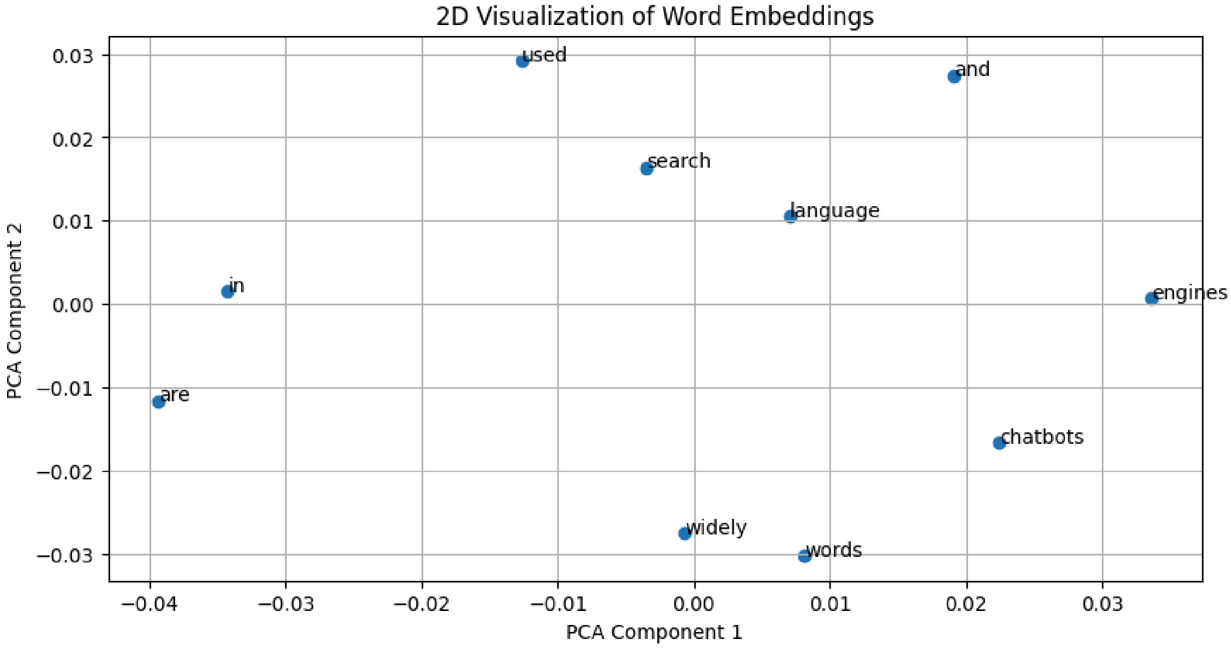
7. Output

- Tokenized corpus
- Word vectors of dimension 100
- 2D scatter plot showing word relationships

---

8. Result

The Word2Vec model successfully generated word embeddings, and PCA visualization showed semantically related words positioned closer in vector space.



9. Conclusion

Word2Vec effectively captures semantic meaning from text data. Dimensionality reduction techniques like PCA help visualize high-dimensional embeddings for better interpretation.

10. Applications

- Text classification
- Search engines
- Chatbots
- Recommendation systems
- Sentiment analysis

11. Future Scope

- Use larger real-world datasets
- Apply advanced embeddings like FastText or BERT
- Perform similarity and analogy tasks
- Integrate embeddings into NLP models

---

12. References

- Gensim Documentation
- NLTK Documentation
- scikit-learn Documentation
- Mikolov et al., “Efficient Estimation of Word Representations in Vector Space”