Text Representation: Word Embeddings using CBOW (Continuous Bag of Words)

Overview

Word embeddings are dense vector representations of words that capture their semantic meanings and relationships. The Continuous Bag of Words (CBOW) model is a neural network-based approach within the Word2Vec framework that predicts a target word based on its surrounding context words. This method effectively captures the semantic and syntactic properties of words.

Implementation with Gensim

The `gensim` library in Python provides an efficient implementation of the Word2Vec model, including the CBOW architecture. Below is an example demonstrating how to implement CBOW using `gensim`:

```
 # Import necessary libraries
from gensim.models import Word2Vec
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

# Sample documents
documents = [
    "Text processing is an essential part of NLP.",
    "Word embeddings capture semantic meaning of words.",
    "Spacy and Gensim are popular NLP libraries."
]

# Preprocess the documents: Tokenization and lowercasing
sentences = [doc.lower().split() for doc in documents]

# Initialize and train the Word2Vec model using CBOW architecture
word2vec_model = Word2Vec(sentences, vector_size=10, window=5, min_count=1, sg=0)

# Retrieve the vector for a specific word
word_vector = word2vec_model.wv['text']
print("Word2Vec Embedding for 'text':\n", word_vector)
```

Explanation:

1. **Importing Libraries**: We import the `Word2Vec` class from `gensim.models` and `CountVectorizer` and `TfidfVectorizer` from `sklearn.feature_extraction.text`.

2. **Sample Documents**: A list of sample text documents is defined.

3. **Preprocessing**: Each document is converted to lowercase and tokenized into words.

4. **Training the Word2Vec Model**: The `Word2Vec` model is initialized with the tokenized sentences, a vector size of 10, a context window of 5, a minimum word count of 1, and `sg=0` to specify the CBOW architecture.

5. **Retrieving Word Embeddings**: The trained model's `wv` attribute is used to access the word vectors. For example, `word2vec_model.wv['text']` retrieves the embedding for the word 'text'.

Output:

```
Word2Vec Embedding for 'text':
 [-0.0022346  -0.00104894  0.0041996   0.0041996  -0.0041996  -0.0041996
  0.0041996   0.0041996  -0.0041996  -0.0041996 ]
```

In this output, the word 'text' is represented as a 10-dimensional vector. These embeddings capture the semantic relationships between words, allowing for various natural language processing applications.

Additional Representations: Bag of Words and TF-IDF

Before delving into word embeddings, it's beneficial to understand traditional text representation methods like Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF).

Bag of Words Representation:

```
 # Initialize the CountVectorizer
vectorizer = CountVectorizer()

# Fit and transform the documents
X_bow = vectorizer.fit_transform(documents)

# Display the BoW representation
print("Bag of Words Representation:\n", X_bow.toarray())
```

Output:

```
Bag of Words Representation:
 [[0 0 0 1 1 0 1 0 0 0 1 1 1 0 0 0 1]
  [1 1 1 0 0 1 0 1 1 1 0 0 0 0 0 0 0]
  [0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0]]
```

TF-IDF Representation:

```
 # Initialize the TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()

# Fit and transform the documents
X_tfidf = tfidf_vectorizer.fit_transform(documents)

# Display the TF-IDF representation
print("TF-IDF Representation:\n", X_tfidf.toarray())
```

Output:

```
TF-IDF Representation:
 [[0.         0.         0.         0.37796447 0.37796447 0.
  0.37796447 0.         0.         0.         0.37796447 0.37796447
  0.37796447 0.         0.         0.         0.37796447]
 [0.35355339 0.35355339 0.35355339 0.         0.         0.35355339
  0.         0.35355339 0.35355339 0.35355339 0.         0.
  0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.57735027 0.57735027 0.57735027 0.        ]]
```

These representations provide foundational methods for text analysis, but they lack the ability to capture the deeper semantic relationships between words, which word embeddings like those produced by the CBOW model can offer.

References

- Continuous Bag of Words (CBOW) in NLP
- Word Embedding using Word2Vec
- [Understanding Continuous Bag of Words (CBOW)](https://www