Analysis of Text Representation Methods in Movie Recommendation Systems

Technical Report

Summary

This report presents a comprehensive analysis of different text representation methods implemented in a movie recommendation system. We examine and compare Bag-of-Words, TF-IDF, Word2Vec, GloVe, and FastText approaches, evaluating their effectiveness in representing movie-related text data and generating recommendations.

1. Comparison of Text Representation Methods

1.1 Implementation Overview

The system was implemented using several key Python libraries:

- pandas and numpy for data manipulation
- sklearn for TF-IDF vectorization and similarity calculations
- gensim for word embedding models
- nltk for text preprocessing
- matplotlib for visualization

1.2 Method Descriptions

Bag-of-Words & TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) extends the basic Bag-of-Words model by weighing terms based on their importance across documents. In our implementation, we used sklearn's TfidfVectorizer with 2000 features, capturing the most significant terms in movie descriptions.

Word2Vec

We utilized Google's pre-trained Word2Vec model (300 dimensions) through gensim. This model captures semantic relationships between words based on their context in a large corpus of text. Word vectors are averaged to create document embeddings.

GloVe

The Global Vectors (GloVe) implementation uses pre-trained embeddings from the Wikipedia/Gigaword corpus. It differs from Word2Vec by focusing on global word co-occurrence statistics rather than local context windows.

FastText

FastText extends Word2Vec by incorporating subword information, making it particularly effective for handling out-of-vocabulary words and morphologically rich languages.

2. Analysis of Pros and Cons

2.1 TF-IDF

Advantages:

- Excellent at capturing document-specific importance of terms
- Computationally efficient
- Works well with domain-specific vocabulary

Limitations:

- Loses word order information
- No semantic understanding
- Sparse representation

2.2 Word2Vec

Advantages:

- Strong semantic understanding
- Dense vector representation
- Good at capturing word similarities

Limitations:

- Fixed vocabulary
- Requires large training corpus
- May lose movie-specific context

- ## 2.3 GloVe

Advantages:

- Captures global statistics effectively
- Balanced semantic and syntactic relationships
- Robust performance on analogy tasks

- **Limitations:**
- Less effective with rare words
- Static embeddings
- Training can be computationally intensive

2.4 FastText

Advantages:

- Handles out-of-vocabulary words well
- Better with rare words due to subword information
- More robust to spelling errors

Limitations:

- Larger model size
- May capture unwanted morphological similarities
- Slower inference time

3. Performance and Recommendations

3.1 Analysis Results

Our implementation tested these methods on the TMDB 5000 movies dataset. Key findings:

1. TF-IDF performed well for exact matching and keyword-based recommendations
2. Word embeddings (Word2Vec, GloVe, FastText) captured semantic similarities better
3. FastText showed superior handling of unique movie titles and rare terms

3.2 Model Selection

FastText emerged as the most effective model for this task due to:

- Better handling of movie-specific terminology
- Robust performance with out-of-vocabulary words
- Good balance of semantic and syntactic understanding

3.3 Implementation Considerations

- Text preprocessing was crucial for all methods
- Combining multiple methods (ensemble approach) could be beneficial

- Regular model updates would be necessary for new movies

4. Conclusion

While each method has its strengths, FastText provides the most robust solution for movie recommendation systems. The combination of subword information and semantic understanding makes it particularly suitable for handling the diverse and evolving nature of movie descriptions.

References

Pre-trained Models

- Word2Vec: Google News (300 dimensions)
- GloVe: Wikipedia + Gigaword corpus
- FastText: Wiki News subwords (300 dimensions)

Libraries

- **pandas** for data manipulation and reading CSV files
- **numpy** for numerical operations and array manipulation
- **scikit-learn** for:
    - cosine similarity implementation (from sklearn.metrics.pairwise import cosine_similarity)
    - TF-IDF implementation (from sklearn.feature_extraction.text import TfidfVectorizer)
- **gensim** for downloading pretrained word embedding models (gensim.downloader as api)
- **matplotlib** for plotting graphs (import matplotlib.pyplot as plt)
- **nltk** for:
    - stopword removal (from nltk.corpus import stopwords)
    - tokenization (from nltk.tokenize import word_tokenize)
    - lemmatization (from nltk.stem import WordNetLemmatizer)
- **json** for parsing and handling JSON data (import json)

Note: *This report reflects findings based on the implementation and testing of these methods on the TMDB 5000 movies dataset. Results may vary with different datasets or use cases.*

Furkan Çoban 202011204