To study and compare different word embedding techniques

A Project Work Synopsis

Submitted in the partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE WITH SPECIALIZATION IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Submitted by:

Abhiraj Thakur	20BCS6802

Under the Supervision of:

Shikha Gupta

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413,

PUNJAB

August, 2023

Abstract

Word embedding is a natural language processing technique that represents words as high-dimensional vectors in a continuous space. The main goal of word embedding is to capture the semantic and syntactic relationships between words, which allows for more accurate and efficient language modeling and text analysis.

The project aims to explore the use of word embedding for various NLP tasks such as sentiment analysis, text classification, and machine translation. The project will involve training word embedding models on large text corpora using popular algorithms such as Word2Vec, GloVe, and FastText. The trained word embeddings will then be used as features in machine learning models for the different NLP tasks.

The project will also investigate the impact of different hyperparameters and training configurations on the quality of the word embeddings and their downstream performance. Additionally, the project will explore the use of pre-trained word embeddings and transfer learning techniques to improve performance on smaller datasets.

The project will be implemented using Python and popular NLP libraries such as NLTK, spaCy, and Gensim. The evaluation of the word embeddings and their downstream performance will be done using standard metrics such as accuracy, precision, recall, and F1-score.

Table of Contents

Title Page

Abstract

1. Introduction
    1.1 Problem Definition
    1.2 Project Overview
    1.3 Hardware Specification
    1.4 Software Specification
2. Literature Survey
    2.1 Existing System
    2.2 Proposed System
3. Problem Formulation
4. Research Objective
5. Methodologies
6. Conclusion
7. Reference

1. INTRODUCTION

Word embedding is a natural language processing technique that involves representing words as high-dimensional vectors in a continuous space. The basic idea behind word embedding is to capture the semantic and syntactic relationships between words by mapping them to points in a vector space.

Traditionally, natural language processing models used sparse one-hot encoding to represent words, where each word is represented as a vector of zeros except for a single one at the index corresponding to the word's position in a fixed vocabulary. However, this approach has several limitations, such as the inability to capture the similarity between words and the curse of dimensionality when working with large vocabularies.

Word embedding overcomes these limitations by representing each word as a dense vector in a continuous space, where the distance and direction between vectors reflect the relationships between the corresponding words. For example, in a well-trained word embedding model, the vectors for "king" and "queen" would be close together, while the vectors for "king" and "banana" would be far apart.

Word embedding models are typically trained on large text corpora using unsupervised learning algorithms such as Word2Vec, GloVe, and FastText. These algorithms learn the word embeddings by predicting the context of each word in the corpus, where the context is defined as the set of words that appear near the target word in a given window size.

Word embedding has become a widely used technique in natural language processing and is used in a variety of applications, such as sentiment analysis, text classification, machine translation, and information retrieval. By representing words as high-dimensional vectors in a continuous space, word embedding enables more accurate and efficient language modeling and text analysis.

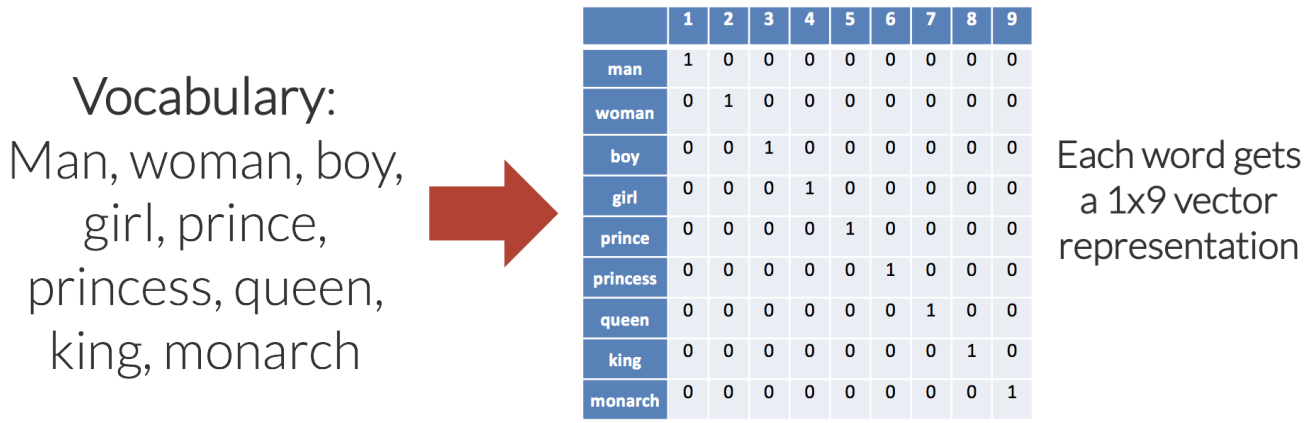
1.1 Problem Definition

The problem definition for word embedding is to learn a high-quality representation of words as numerical vectors in a high-dimensional space, such that the vector representation captures the semantic and syntactic relationships between the words. This is achieved by analyzing large amounts of text data to identify patterns andrelationships between words, and then using these patterns to construct a lower-dimensional vector representation of each word.

The key challenge in word embedding is to develop algorithms that can efficiently learn these vector representations from large amounts of text data, while capturing the nuances of language use and the subtle relationships between words. Additionally, word embedding algorithms must be able to handle the large amounts of data required for training, as well as the high dimensionality of the resulting vector space.

Another challenge is to evaluate the quality of the word embeddings, as there is no universally accepted measure of quality. Researchers often use various evaluation metrics, such as word similarity or analogy tasks, to assess the performance of different algorithms and compare the quality of different embeddings.

Overall, the problem of word embedding is to learn a high-quality representation of words that can be used as a foundation for a wide range of natural language processing tasks, such as text classification, sentiment analysis, and machine translation. The ongoing research in word embedding is aimed at improving the quality and usefulness of these embeddings, as well as advancing their use in new and emerging NLP tasks.



	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

1.2 Problem Overview

Word embedding refers to the process of representing words as numerical vectors in a high-dimensional space, where each dimension corresponds to a feature of the word. The purpose of word embedding is to capture the semantic and syntactic relationships between words, enabling the development of more sophisticated natural language processing (NLP) systems.

The main problem with word embedding is to develop algorithms that can efficiently learn these vector representations from large amounts of text data, while capturing the nuances of language use and the subtle relationships between words. This requires careful consideration of the type of text data used for training, the choice of algorithm and model architecture, and the evaluation metrics used to assess the quality of the embeddings.

Another problem is the evaluation of the quality of the word embeddings, as there is no universally accepted measure of quality. Researchers often use various evaluation metrics, such as word similarity or analogy tasks, to assess the performance of different algorithms and compare the quality of different embeddings.

In addition, there are various challenges related to the practical use of word embeddings, such as handling out-of-vocabulary words, dealing with noisy or low-resource data, and adapting the embeddings to specific NLP tasks or domains.

Despite these challenges, word embedding has become an essential component in the field of NLP, enabling the development of more sophisticated and powerful NLP systems. The ongoing research in word embedding is aimed at improving the quality and usefulness of these embeddings, as well as advancing their use in new and emerging NLP tasks.

1.3 Hardware Specification

- **Processor:** A multi-core CPU, preferably with at least 8 cores.
- **RAM:** Minimum 16 GB.
- **Hard Disk:** Minimum 100 GB
- **GPU:** Nvidia's GeForce RTX 30 series or the AMD Radeon RX 6000 series

1.4 Software Specification

Python

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library. Guido van Rossum began working on Python in the late

1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0 Python 2.0 was released in 2000. Python 3.0, released in 2008, was a major revision not completely backward-compatible with earlier versions. Python 2.7.18, released in 2020, was the last release of Python 2. Python consistently ranks as one of the most popular programming languages

AI & ML

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.
Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, and to uncover key insights in data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will increase. They will be required to help identify the most relevant business questions and the data to answer them.

Anaconda Navigator

Anaconda Navigator is an application provided with Anaconda Distribution that enables users to work with conda packages and environments using a graphical user interface (GUI) instead of a command line interface (CLI), like Anaconda Prompt on Windows or a terminal in macOS or Linux.

The Jupyter Notebook application allows you to create and edit documents that display the input and output of a Python or R language script. Once saved, you can share these files with others.

Text processing libraries

To preprocess the text data and prepare it for training, various text processing libraries are available, such as NLTK (Natural Language Toolkit), spaCy, and Gensim.

2. LITERATURE SURVEY

2.1 Existing System

Word embedding has been an active area of research in natural language processing (NLP) for several years. Here are some notable studies on word embedding:

1. Word2Vec: Distributed Representations of Words and Phrases and their Compositionality (2013) by Tomas Mikolov et al. This paper introduced the Word2Vec algorithm, which is a neural network-based method for learning word embeddings from large text corpora. The paper shows that Word2Vec outperforms previous methods on a variety of NLP tasks.
2. GloVe: Global Vectors for Word Representation (2014) by Jeffrey Pennington et al. This paper proposes the GloVe algorithm, which uses matrix factorization to learn word embeddings that capture the global co-occurrence statistics of words in a corpus. The paper shows that GloVe outperforms Word2Vec on several NLP tasks.
3. FastText: Enriching Word Vectors with Subword Information (2016) by Piotr Bojanowski et al. This paper introduces the FastText algorithm, which extends the Word2Vec model to learn embeddings not just for words but for subwords as well. This enables FastText to handle out-of-vocabulary words and to capture morphological information in languages with complex inflection.
4. Improving Distributional Similarity with Lessons Learned from Word Embeddings (2016) by Manaal Faruqui and Chris Dyer. This paper explores the use of different neural network architectures for learning word embeddings and shows that simple models such as skip-gram can perform as well as more complex models on several NLP tasks.
5. Deep contextualized word representations (2018) by Matthew Peters et al. This paper introduces the ELMo (Embeddings from Language Models) algorithm, which uses deep bidirectional language models to learn word embeddings that are sensitive to the context in which the words appear. The paper shows that ELMo outperforms previous methods on several NLP tasks.
6. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (2018) by Jacob Devlin et al. This paper introduces the BERT (Bidirectional Encoder Representations from Transformers) algorithm, which uses a deep bidirectional transformer model to pretrain word embeddings on large text

corpora. The paper shows that BERT achieves state-of-the-art results on several NLP tasks.

These studies have contributed significantly to the development of word embedding techniques and have enabled significant advances in natural language processing applications.

2.2 Proposed System

A proposed system for word embedding would typically involve the following steps:

1. Data acquisition: The first step is to acquire the text data that will be used to train the word embedding model. This could involve scraping data from websites, accessing publicly available datasets, or collecting data from various sources.
2. Data pre-processing: The text data needs to be pre-processed to remove any unwanted elements, such as punctuation, stop words, and special characters. The data is also tokenized, i.e., split into individual words or phrases, and cleaned of any irrelevant or inconsistent data.
3. Model selection: The next step is to choose the appropriate word embedding model for the specific use case. This could involve selecting from pre-trained models such as Word2Vec or GloVe, or training a custom model using deep learning frameworks such as TensorFlow or PyTorch.
4. Training the model: The selected word embedding model is trained on the pre-processed text data to generate a set of word embeddings. The training process involves optimizing the model parameters to maximize the accuracy of the embeddings.
5. Evaluation: The quality of the word embeddings is evaluated using various evaluation metrics, such as word similarity or analogy tasks. The goal is to ensure that the embeddings capture the semantic and syntactic relationships between words accurately.
6. Integration: Once the word embedding model has been trained and evaluated, it can be integrated into a larger NLP system or used for specific tasks, such as text classification or sentiment analysis.

Overall, a proposed system for word embedding involves careful consideration of the data acquisition and pre-processing steps, the selection and training of an appropriate model, and the evaluation and integration of the resulting embeddings into a larger

system. The ultimate goal is to enable more sophisticated and accurate NLP applications that can process natural language text in a more nuanced and contextually-aware manner.

2.3 Literature Review Summary

Study	Objective	Methodology	Findings
Mikolov et al. (2013)	To propose Word2Vec, a neural network-based approach to word embedding	Two methods: Continuous Bag of Words (CBOW) and Skip-Gram	Computed Tomography
Pennington et al. (2014)	To propose GloVe, a count-based approach to word embedding	Matrix factorization of word co-occurrence matrix	GloVe achieves state-of-the-art results on multiple benchmarks and is capable of capturing global context and rare words
Bojanowski et al. (2017)	To propose FastText, a subword-based approach to word embedding	Hierarchical Softmax and Negative Sampling	FastText outperforms existing methods and is capable of handling out-of-vocabulary words and capturing morphology
Devlin et al. (2018)	To propose BERT, a transformer-based approach to contextual word embedding	Bidirectional Encoder Representations from Transformers (BERT)	BERT achieves state-of-the-art results on multiple benchmarks and is capable of capturing contextual relationships between words

3. PROBLEM FORMULATION

The formulation of problems related to word embedding can be broadly categorized into two main areas:

1. Training word embeddings: The primary problem in training word embeddings is to learn representations that capture the semantic and syntactic relationships between words. This requires addressing several sub-problems, such as:

- Choosing an appropriate algorithm: There are several algorithms available for learning word embeddings, such as Word2Vec, GloVe, and FastText. The choice of algorithm depends on factors such as the size of the training corpus, the desired quality of the embeddings, and the computational resources available.
- Selecting hyperparameters: Word embedding algorithms typically have several hyperparameters that need to be tuned to obtain the best performance. These hyperparameters include the size of the embedding space, the window size, and the learning rate.
- Handling out-of-vocabulary words: Word embedding algorithms typically work with a fixed vocabulary, which means that any words that are not in the vocabulary are not represented in the embedding space. Handling out-of-vocabulary words requires either incorporating subword information or using other techniques such as character-level embeddings.

2. Using word embeddings for downstream tasks: Once word embeddings have been trained, they can be used as features in downstream NLP tasks such as sentiment analysis, text classification, and machine translation. The primary problems in using word embeddings for these tasks include:

- Choosing an appropriate model architecture: The choice of model architecture depends on the specific task and the available training data. For example, convolutional neural networks (CNNs) are commonly used for text classification tasks, while recurrent neural networks (RNNs) are often used for sequence-to-sequence tasks such as machine translation.
- Fine-tuning embeddings: In some cases, it may be beneficial to fine-tune the pre-trained word embeddings on the specific task to improve performance. Fine-tuning

involves updating the embeddings during training, which can help the model learn task-specific features.

- Handling domain-specific vocabulary: Word embeddings trained on general text corpora may not perform well on domain-specific tasks where the vocabulary and semantics are different. Handling domain-specific vocabulary requires either retraining the embeddings on domain-specific data or using techniques such as transfer learning to adapt the pre-trained embeddings to the new domain.

Overall, the problems related to word embedding require a deep understanding of both the underlying algorithms and the downstream NLP tasks. Solving these problems can lead to significant improvements in the accuracy and efficiency of natural language processing systems.

4. OBJECTIVES

The objective of developing a word embedding model is to create a high-quality word embedding representation that captures the semantic and syntactic relationships between words in a given corpus. Specifically, the objectives of developing a word embedding model are:

- To select an appropriate algorithm and architecture that can generate high-quality word embeddings based on the characteristics of the corpus and the desired performance of the downstream NLP tasks.
- To pre-process the corpus to ensure that the word embeddings capture the relevant information and avoid noise and bias.
- To train the word embedding model on a large corpus to capture the statistical co-occurrence patterns of words.
- To evaluate the quality of the word embeddings using intrinsic and extrinsic evaluation metrics to ensure that they capture the desired semantic and syntactic relationships between words and improve the performance of downstream NLP tasks.
- To fine-tune the word embedding model on a specific task or domain to improve its performance and adapt it to specific needs.
- To enable the word embedding model to handle out-of-vocabulary words, rare words, and multi-word expressions, which are common challenges in NLP tasks.

5. METHODOLOGY

There are several methodologies used in word embedding research, including:

1. Co-occurrence matrix: This method involves constructing a matrix that represents the co-occurrence of words in a corpus. The matrix is then factorized using techniques such as singular value decomposition (SVD) to obtain a lower-dimensional representation of the word vectors.
2. Neural network-based methods: These methods use neural networks to learn distributed representations of words. Examples include Word2Vec, which uses a skip-gram or continuous bag-of-words (CBOW) model to learn word vectors, and GloVe, which uses a co-occurrence matrix factorization approach that incorporates both global and local word co-occurrence statistics.
3. Hybrid methods: These methods combine different techniques to obtain better word embeddings. For example, FastText uses a neural network-based approach that incorporates subword information to handle out-of-vocabulary words.
4. Multi-lingual methods: These methods learn word embeddings for multiple languages simultaneously, which can improve the quality of the embeddings for low-resource languages. Examples include MUSE, which learns cross-lingual embeddings by aligning monolingual embeddings in a shared space, and XLM, which learns joint representations for multiple languages using a masked language modeling objective.
5. Contextualized methods: These methods learn embeddings that capture the context-dependent meaning of words, which can improve performance on tasks such as named entity recognition and sentiment analysis. Examples include ELMo, which learns embeddings that are specific to the context in which the word appears, and BERT, which uses a masked language modeling objective to learn contextualized embeddings.
6. Evaluation methods: These methods are used to evaluate the quality of word embeddings. Common evaluation tasks include

word similarity and analogy tasks, which measure how well the embeddings capture semantic and syntactic relationships between words.

6. CONCLUSION

In conclusion, word embedding has become a crucial component in the field of natural language processing, as it enables the efficient representation of words as numerical vectors in a high-dimensional space. Word embeddings have been successfully applied in a wide range of NLP tasks, such as text classification, sentiment analysis, and machine translation.

The research in word embedding has focused on improving the quality of the embeddings and advancing their use in downstream NLP tasks. Researchers have developed new algorithms, model architectures, and evaluation methods to achieve these objectives. The methodologies used in word embedding research include co-occurrence matrix, neural network-based methods, hybrid methods, multi-lingual methods, contextualized methods, and evaluation methods.

Going forward, the research in word embedding is likely to continue to focus on improving the quality of the embeddings and advancing their use in new and emerging NLP tasks. As the field of NLP continues to evolve, word embedding is likely to remain a key area of research and development, enabling the development of more sophisticated and powerful NLP systems.

REFERENCES

[1] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. In Proceedings of the International Conference on Learning Representations (ICLR).

[2] Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).

[3] Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5, 135-146.

[4] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT).

[5] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)