**Word Embeddings using Machine Learning**

Author: Jose Manuel Alvarez Gonzalez

Email Id: jmanuel.ag89@gmail.com

Date: 20/11/2023

Contents

[Introduction 3](#_Toc151374298)

[Word Embeddings 3](#_Toc151374299)

[Data Preparation 4](#_Toc151374300)

[Word2vec 5](#_Toc151374301)

[Skip-gram 6](#_Toc151374302)

[CBOW 6](#_Toc151374303)

[Gensim 7](#_Toc151374304)

[Benchmark Model 7](#_Toc151374305)

[Text Classification 7](#_Toc151374306)

[Data Processing 8](#_Toc151374307)

[Analysis of Results 9](#_Toc151374308)

[Conclusions 9](#_Toc151374309)

[Further Improvements 9](#_Toc151374310)

[Future on Word Embeddings 10](#_Toc151374311)

[Appendix A – Data Links 11](#_Toc151374312)

[Appendix B – References 11](#_Toc151374313)

Introduction

In this project, we dive into the realm of Natural Language Processing (NLP), focusing on the construction of a word embedding – a numerical representation of words – using the word2vec model. This model, introduced in the seminal 2013 paper "Efficient Estimation of Word Representations in Vector Space," utilizes a linear neural network for embedding generation. Our dataset comprises 108,414 movies from IMDB, with a focus on their titles, genres, and descriptions, the latter being crucial for our model building.

Word embeddings are pivotal in NLP for capturing semantic relationships between words. The word2vec model, known for its efficiency and effectiveness, will be employed to transform textual data into meaningful vector representations. We will detail the preprocessing steps, the architecture of the neural network, and the training process used to develop these embeddings.

A critical aspect of our study involves comparing the embeddings generated by word2vec with the pre-trained GloVe (Global Vectors for Word Representation) embeddings. This comparison will be based on their performance in a text classification task, for which we will employ a Recurrent Neural Network. This approach allows us to assess the relative strengths of our model in capturing linguistic nuances.

Word Embeddings

Word embeddings are a numerical representation of a word in a vector space, enabling their computation for numerical tasks. Besides representing words as numbers and reduce dimensionality from other approaches, they also capture semantic and syntactic similarities.   
  
**Brief History**

* **One-Hot Encoding**: can be seen as the earliest representation in a numerical form for a word, this approach requires a matrix of size NxN, where N is the total size of the vocabulary. This approach is highly dimensional and doesn’t capture any relationship about the word.
* **Term Frequency-Inverse Document Frequency (TF-IDF):** A significant improvement over one-hot encoding, TF-IDF weighed words based on their frequency and uniqueness across documents, offering a more informative representation.
* **Latent Semantic Analysis (LSA):** Introduced in the late 1980s, LSA utilized techniques like singular value decomposition on word-context matrices to reduce dimensionality and unearth latent semantic relationships.

* **Neural Network-based Models:** The early 2000s marked the rise of neural network-based models for embedding generation. 2003 paper from Bengio "A Neural Probabilistic Language Model" was a pioneering work in this regard.
* **Word2vec:** Published by Mikolov et al. in 2013, is a groundbreaking framework for learning word vectors. It operates based on the context surrounding a given word, significantly improving the efficiency and quality of word embeddings.

Word2vec was a pivotal step in the development of NLP, this is what made me focus my research on understanding how they operate.

Data Preparation

The IMDB dataset chosen for the project was download from a Kaggle repository, link available on the appendix. The dataset comprises 108,414 movies, with a focus on their titles, genres, and descriptions. IMDB movie datasets can be found across multiple NLP research projects:

* 2016 Research by Zhang et al., "Character-level Convolutional Networks for Text Classification": Utilizing the IMDb dataset, this study explored the effectiveness of character-level convolutional networks in text classification, marking a shift from traditional word-level processing.
* 2020 Research by Chun-Liang et al. "Sentiment Analysis of IMDB Movie Reviews with NLP" - This research applies NLP techniques for sentiment analysis on the large collection of movie reviews from the IMDb dataset.
* 2020 Research by Qaisar et al. “Sentiment Analysis of IMDb Movie Reviews Using Long Short-Term Memory.”

Like any other machine learning model, textual data also requires some prepossessing like cleaning and normalizing. Text processing is done slightly differently, we also followed two stages of processing for building training data, the first prior building the word2vec model and the second prior building the text classification.

The following are the steps followed to initially process the text corpus, which is the movie description for each film:

* Collected the description of the 10 most used movie genres and evenly distribute the data, with 1,550 samples for each genre.
* Convert text to lower case.
* Expand contractions.
* Remove special characters and numbers.
* Remove stop words.
* Tokenization, the process of separating each word of the sentence as an individual item.
* Lemmatization, the process of converting each word to the root base, this reduces the number of words.

After processing our text, we create a full vocabulary for all the words found in the corpus. The finial corpus includes a total of 49,196 different words, with an average of 55 words per movie description. For this project I decided to keep each word, as the last stage of the processing we create a dictionary giving each word a number reference (“word”: 1).

Word2vec

The original paper of the word2vec framework published in 2013, this framework is based on the idea that a word’s meaning is defined by its context. Context is represented as the surrounding words, given a specific word window.

The model iterates across each sentence, focusing on a target word and the context across the word making predictions for either the target word or the context words.

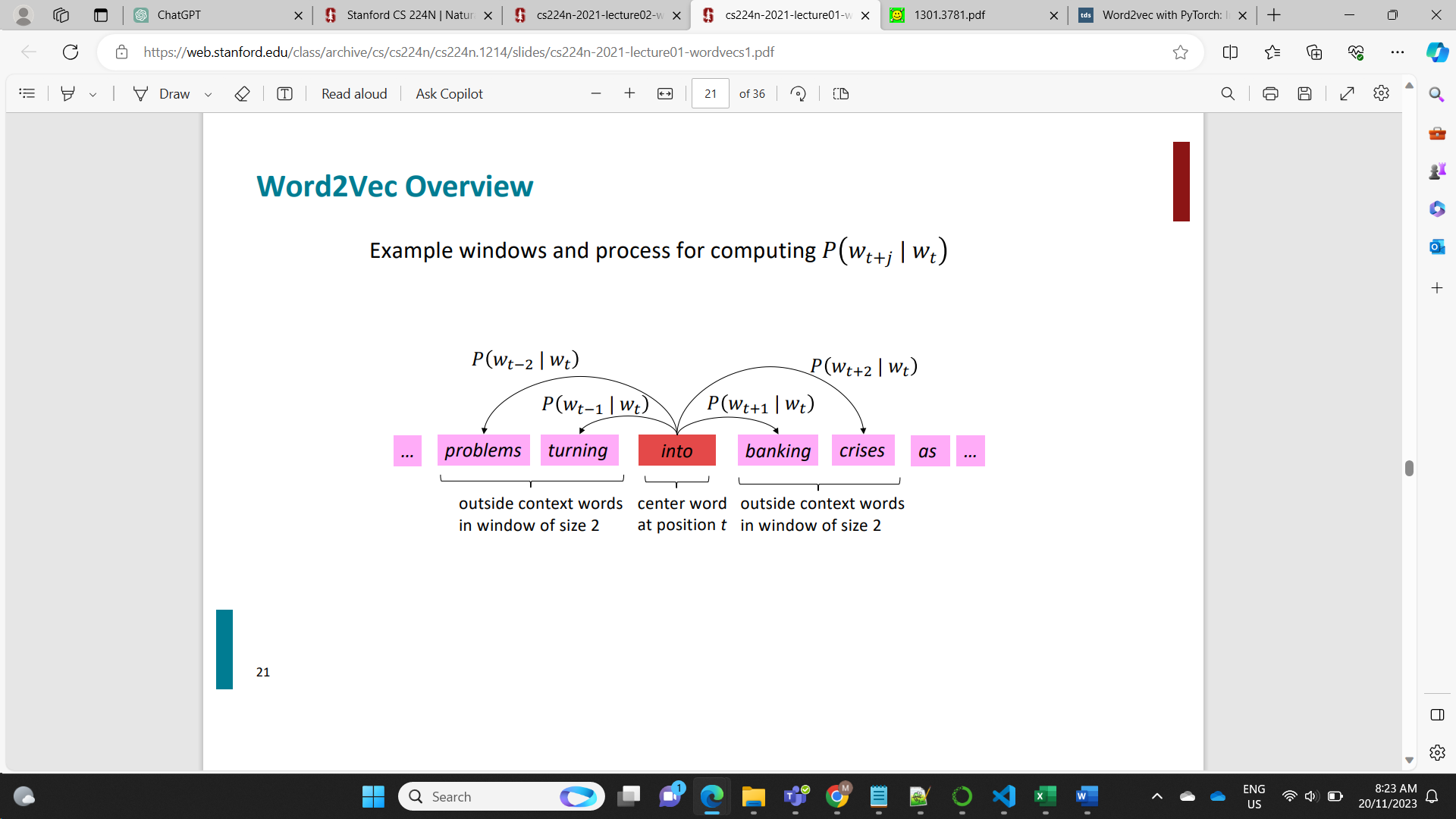


Figure 1, predicting context from a target word. Stanford University. (2021). CS224N: Natural Language Processing with Deep Learning [PDF file]. Retrieved from https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1214/slides/cs224n-2021-lecture01-wordvecs1.pd

The model can be build using two approaches:

* CBOW: Takes the context words as inputs and tries to predict the target word.
* Skip-gram: Takes a target word and tries to predict context words.

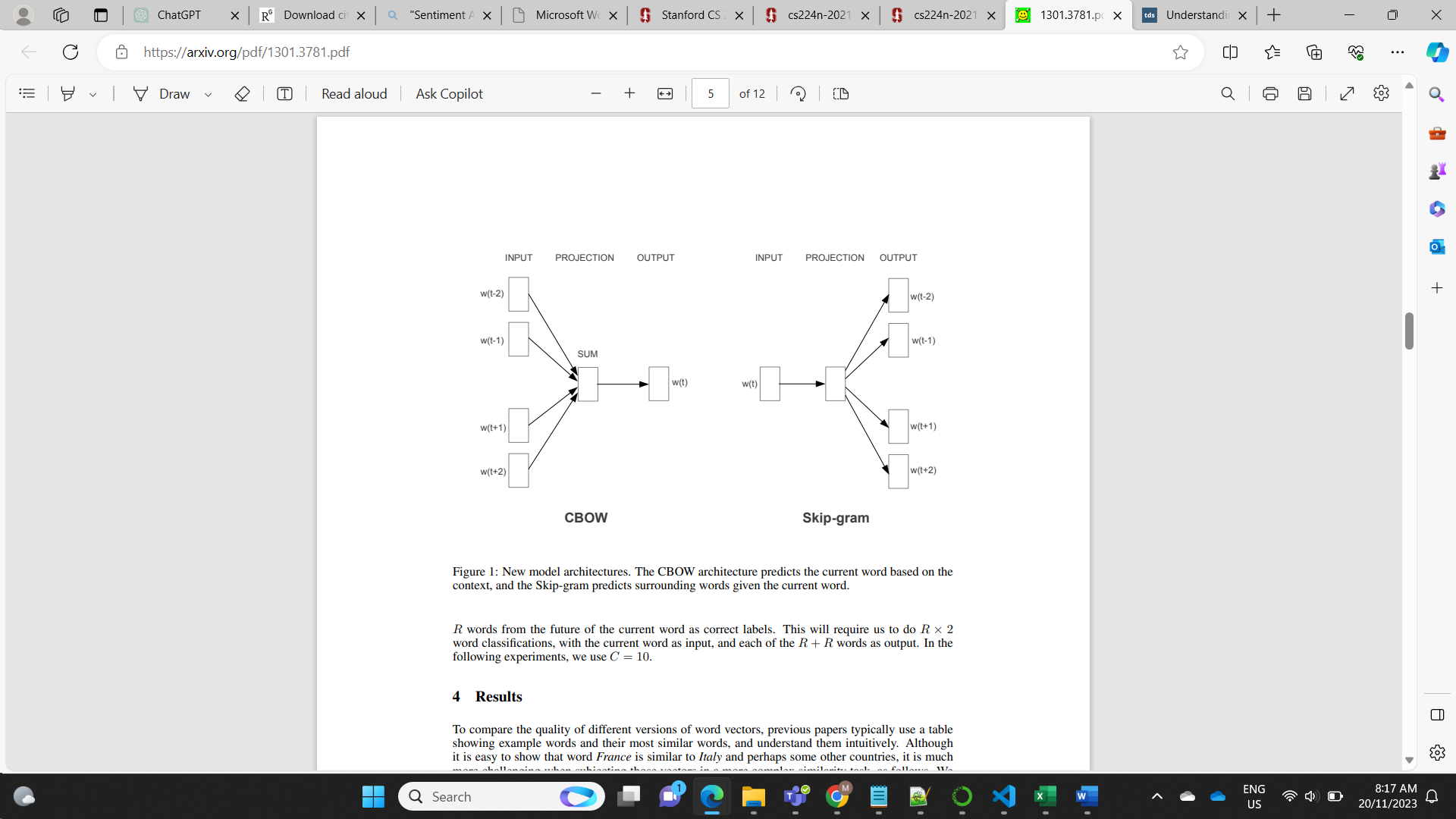


Figure 2, approaches for word2vec. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781.

The prediction is done using a neural network architecture. For this project, the architectures used, comprises of two main components: an embedding layer and a linear layer. The embedding layer converts input (target or context) words into 100-dimensional vectors, capturing their semantic properties. Following this, the linear layer maps these embeddings back to the vocabulary space, predicting output words (target or context). This compact architecture enables the model to learn meaningful word relationships based on contextual usage.

For this project we created three different word2vec models and used a benchmark model, described on the next section.

## Skip-gram

This model takes a context word and makes a prediction for the target word. The first part of this stage comprises on building the training data. For this propose, we took a context window of two words.   
  
For the training data, we took one word as input and one word as context output from the predefined context window. From the initial corpus, we finished with a total of 3’468,392 different samples to feed the network. I decided to use them all.   
  
To feed the training data to the network, we created a dataloader with a batch size of 128 which shuffles the data. The training step, had the following parameters:

* Loss function: Cross Entropy.
* Optimizer: Stochastic Gradient Decent
* Epochs: 20
* Learning Rate: 0.01 first 10 epochs, 0.001 for next 5 epochs, 0.0001 last 5.

After finished the training, we retrieve the embeddings of each word from the vocabulary, as this is the main propose of the model rather than accuracy of prediction.

## CBOW

This model followed the same training parameters and network architecture used from the skip-gram model. The main difference in this model is the training data used. For this data, we used a context window of two, but in this case, we only kept a sample when the four context words where present around the target.   
  
After processing, we finished with a total of 828,348 different samples. Each sample had four context words as input and one target word as output. After training, we collected the embeddings of the model.

## Gensim

Gensim, an open-source Python library released in 2010 by Radim Řehůřek, is popular for its implementation of word2vec, it allows to build your own word2vec in one simple function.

I utilized Gensim's word2vec implementation to create our third model on the same corpus as the last two. The training was conducted over 20 epochs, employing the default parameters.

Benchmark Model

The benchmark model for comparing our word2vec embeddings is GloVe (Global Vectors for Word Representation), an unsupervised learning algorithm designed to obtain vector representations of words. GloVe builds these representations by analyzing global word-word co-occurrence statistics from a given corpus, resulting in word vectors that reveal intriguing linear substructures in the word vector space.

For our project, we utilized a GloVe model that was trained on a 2014 Wikipedia corpus. This training involved 6 billion tokens and resulted in a vocabulary of 400,000 unique words. The vectors in this model have a dimensionality of 100.

To ensure comprehensive coverage of our corpus in the classification tests, we supplemented the GloVe model with embeddings from the Gensim model for tokens in our corpus that were not represented in the GloVe embeddings. This approach is akin to transfer learning, where we leverage the GloVe embeddings and enhance them with the word representations learned from our specific corpus using Gensim.

# Text Classification

This is the stage where we are going to compare our three different word embeddings against the benchmark model. The classification will use a Recurrent Neural Network to predict the movie genre, based on the movie description using the word embeddings as inputs on the sequence.

Before we continue, let’s look at a brief history of RNN for text classification:

* **1988 - Stephen Grossberg's Paper on RNNs**: Published foundational work on RNNs, providing early insights into neural networks capable of processing sequence data.
* **1997 - Introduction of LSTM by Hochreiter and Schmidhuber:** LSTM addresses the vanishing gradient problem inherent in traditional RNNs.
* **2010 - The Renaissance of Neural Networks:** The 2010s marked a resurgence in neural network research, with RNNs, including LSTMs.
* **2014 - Sequence to Sequence Learning:** The development of sequence-to-sequence learning models by Sutskever, Vinyals, and Le, which often utilized RNNs, with complex applications such as machine translation.
* **2017 - Introduction of Transformers:** The Transformer model was introduced in a paper titled "Attention Is All You Need" by Vaswani et al. in 2017. This architecture, based on self-attention mechanisms and the main model currently use for NLP.

For the classification we are going to use a vanilla RNN, the model is structured to accept 100 dimensional vectors as inputs, aligning with our word embeddings, and targets a 10 category output corresponding to different genres. The initial recurrent layer with two levels. The RNN's initial hidden state is zero-initialized for each data batch. Post recurrent layer processing, the final hidden state is fed into a linear layer, mapping it to the 10-dimensional output space without use of activation function.

## Data Processing

For the first stage of processing, I looked at the length of each movie description and decided to keep only movies with 10 up to 100 tokens on their descriptions. This stage left a total of 1,237 samples for each genre. We continue with our splits ensuring that each class was equally divided:

* Training 50%
* Validation 30%
* Test 20%

For the inputs, I also utilized a technique called padding which adds a tensor of “0’s” to ensure all the descriptions are of equal length. Here I also found an important hyperparameter for our RNN model, which is the total length of the description to use in the classification. I found that if we use the total length of 100, we encounter a vanishing gradient problem and we can’t accurately predict.

I limited the descriptions length to only 25 tokens giving us the best results. These transformations were made with a custom Dataset class and feed in to a Dataloader of batch size 32.

Analysis of Results

All the word embeddings where use in the RNN model for the classification, using the same data and the following training parameters:

* Loss function: Cross Entropy.
* Optimizer: Stochastic Gradient Decent
* Epochs: 40
* Learning Rate: 0.01 first 20 epochs, 0.001 for next 10 epochs, 0.0001 last 10.

This training and further evaluation on test data generated the following table of results, the measurements are based on the accuracy of gender prediction:



Figure 3, Results of classification task

## Conclusions

* The overall best performance on test data was GloVe-100 with 46.14% accuracy followed by Gensim with 45.58%.
* For our own models, CBOW performed better than Skip-gram with 18.47% over 17.82%.
* All models except GloVe clearly overfitted their models with almost a 20% difference between training and validation.
* Gensim performed almost like GloVe, this might be a limitation of the RNN.
* For a less complex and target specific, Gensim approach can be the most efficient.

## Further Improvements

* Reduce the vocabulary to most common words.
* Test different embedding dimensions.
* Use a regularization on the network training.
* For the classification model, try another architecture like LSTM to be able to include longer movie descriptions.

Future on Word Embeddings

* **2015 - FastText by Facebook:** Developed by Joulin et al., FastText extended the word2vec idea to consider subword information (like character n-grams), allowing it to generate better embeddings for rare words.
* **2018 - ELMo (Embeddings from Language Models):** Introduced by Peters et al., ELMo represented a shift towards context-dependent embeddings, generating different embeddings for a word based on its contextual use.
* **2018 - BERT (Bidirectional Encoder Representations from Transformers):** Developed by Devlin et al. at Google, BERT revolutionized the field by introducing a deeply bidirectional, unsupervised language representation, pretrained using only a plain text corpus.
* **2019 to Transformer-based Models:** Many models like GPT (Generative Pre-trained Transformer), RoBERTa (Robustly optimized BERT approach), T5 (Text-To-Text Transfer Transformer), created their own transformer-based embeddings focusing on contextually rich and multiple dimensions.

# Appendix A – Data Links

Genre Classification Dataset - IMDb. (2023). [Data set]. Retrieved November 18, 2023, from https://www.kaggle.com/datasets/hijest/genre-classification-dataset-imdb/data

Stanford University. (n.d.). GloVe: Global Vectors for Word Representation. Retrieved November 18, 2023, from <https://nlp.stanford.edu/projects/glove/>

# Appendix B – References

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. Journal of Machine Learning Research, 3, 1137-1155. Retrieved from https://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805. Retrieved from https://arxiv.org/abs/1810.04805

Grossberg, S. (1988). Nonlinear neural networks: Principles, mechanisms, and architectures. Neural Networks, 1(1), 17-61. Retrieved from http://techlab.bu.edu/files/resources/articles\_cns/grossberg1988.pdf

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780. Retrieved from https://www.researchgate.net/publication/13853244\_Long\_Short-term\_Memory

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of Tricks for Efficient Text Classification. arXiv preprint arXiv:1607.01759. Retrieved from https://arxiv.org/abs/1607.01759

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1532-1543).

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. arXiv preprint arXiv:1802.05365. Retrieved from https://arxiv.org/abs/1802.05365

Qaisar, S. (2020). Sentiment Analysis of IMDb Movie Reviews Using Long Short-Term Memory. IEEE. doi:10.1109/ICCIS49240.2020.9257657

Rehurek, R., & Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks (pp. 45-50). Valletta, Malta: ELRA. Retrieved from http://is.muni.cz/publication/884893/en.

Stanford University. (n.d.). CS224N: Natural Language Processing with Deep Learning. Retrieved November 18, 2023, from https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1214/

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems (pp. 3104-3112). Retrieved from https://arxiv.org/abs/1409.3215

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In Advances in Neural Information Processing Systems (pp. 5998-6008).

Wu, Chun-Liang & Shin, Song-Ling (2020). Machine learning based classification for sentimental analysis of IMDb reviews. Retrieved from https://cs229.stanford.edu/proj2020spr/report/Wu\_Shin.pdf

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. arXiv preprint arXiv:1509.01626. Retrieved from https://arxiv.org/abs/1509.01626