NLP Text Classification

CMPE/CISC 452

Andy Craig (20146870)
Devynn Garrow (20148909)
Jessica Guetre (20145059)
Tom Hamilton (20165886)

Group 13

Prof. Hazem Abbas

December 6th, 2023

Table of Contents

1 Motivation 4

2 Problem Description 4

2.1 Dataset 4

2.2 GloVe 5

2.3 Word2Vec 5

2.4 Evaluation Framework 5

2.5 Challenges and Limitations 5

3 Contributions 5

4 Related Work 6

4.1 Foundational Paper* 6

4.2 Relevant Journal Paper* 7

4.3 Paper Addressing Similar Issue* 7

5 Datasets 7

6 Implementation 8

6.1 Data Preprocessing 8

6.2 Experimental Setup 9

6.3 Model Architecture 10

6.4 Training, Testing, and Validation 11

7 Results and Discussion 12

7.1 Results 12

7.2 Discussion 12

8 Conclusion and Future Work 16

8.1 Areas for Improvement: 16

9 Work Cited 17

List of Tables

Table 1: Group members and their individual contributions. _____ 5
Table 2: 20NewsGroups categories partitioned by subject matter [5]. _____ 8
Table 3: Sample of the comp.graphics newsgroup from the 20NewsGroups dataset [4] _____ 8
Table 4: Sample of the comp.graphics newsgroup from the 20NewsGroups dataset refined by the preprocessing code [4]. _____ 8
Table 5: Sample of the comp.graphics newsgroup from the 20NewsGroups dataset refined by the preprocessing code [4]. _____ 9
Table 6: Results from implemented architectures and published models [1] to compare pre-trained word embeddings and classification models. Errors were obtained using results from 3 runs. Note that the last three columns were unavailable from the published models, thus are listed as N/A. _____ 12

List of Figures

Figure 1: Dataflow diagram depicting the entire preprocessing stage. _____ 9
Figure 2: CNN model architecture. _____ 11
Figure 3: LSTM model architecture. _____ 11
Figure 4: Architecture of CNN + LSTM model _____ 11
Figure 5: Classification Model Comparison of Validation Accuracy for GloVe Pre-trained Word Embeddings _____ 13
Figure 6: Classification Model Comparison of Validation Accuracy for Word2Vec Pre-trained Word Embeddings _ 14
Figure 7: Embedding-independent Classification Model Comparison of Validation Accuracy _____ 15
Figure 8: Model-independent Embedding Comparison of Validation Accuracy _____ 15

1 Motivation

Our motivation to explore NLP text classification arises from the impactful study, "A Comparative Study on Word Embeddings in Deep Learning for Text Classification," examined in the literature review. Conducted on the 20NewsGroups dataset, the study highlights the crucial role of word embeddings in text classification, comparing classical and contextual embeddings to guide optimal model selection.

The current surge of interest in NLP and genetic algorithms, evident in innovations like ChatGPT and ML applications in natural language processing, has driven this exploration. We are intrigued by how these generative algorithms process and interpret natural language, motivating us to investigate their potential applications in text classification. This exploration is pivotal, potentially shaping the future of text-processing mechanisms and providing insights into the effectiveness of different embedding strategies.

Our project aims to reveal the nuanced performance of classical versus complex word embeddings, recognizing that classical embeddings might outperform their more intricate counterparts. This comparative analysis seeks to offer valuable insights into the evolving landscape of NLP, guiding the development of future text classification models. Optimizing word embeddings for text classification is critical in practical applications, ranging from information retrieval systems to sentiment analysis tools. Our project directly addresses this by comparing classical and contextual embeddings on the widely used 20NewsGroups dataset.

The deliberate choice of the 20NewsGroups dataset for the experiments is grounded in its ubiquity sin the field and its inclusion in the reference paper. It is a pertinent benchmark for comparing various embedding approaches in text classification, aligning with the objective to contribute meaningfully to ongoing NLP discussions.

2 Problem Description

Our project addresses a fundamental question in natural language processing (NLP): determining the optimal combination of word embeddings and neural architectures for effective text classification. This exploration begins by comparing the performance of two widely utilized pre-trained word embeddings, namely GloVe and Word2Vec, across three distinct neural architectures—Convolutional Neural Network (CNN), Long Short-Term Memory Network (LSTM), and a hybrid of CNN with LSTM.

2.1 Dataset

The dataset selected is the well-established 20NewsGroups dataset. Comprising around 20,000 newsgroup documents categorized into 20 distinct groups, 20NewsGroups stands as a cornerstone in the realm of machine learning applied to text. Renowned for its use in text classification and clustering experiments, this dataset serves as a robust foundation for the comparative analysis. This project delves into two types of word embeddings to discern their impact on text classification, GloVe and Word2Vec.

2.2 GloVe

GloVe operates as an unsupervised learning algorithm, generating vector representations for words based on global word-word co-occurrence statistics. Our choice of GloVe is grounded in its ability to reveal interesting linear substructures within the word vector space.

2.3 Word2Vec

Word2Vec represents a family of model architectures and optimization techniques for learning word embeddings from extensive datasets. We opt for Word2Vec due to its proven success across various downstream NLP tasks, showcasing its versatility.

2.4 Evaluation Framework

Our evaluation framework involves six distinct combinations of word embeddings and model architectures:

1. GloVe + CNN
2. GloVe + LSTM
3. GloVe + CNN + LSTM
4. Word2Vec + CNN
5. Word2Vec + LSTM
6. Word2Vec + CNN + LSTM

Effectiveness will be gauged by the model's accuracy in correctly classifying text segments into the relevant news categories.

2.5 Challenges and Limitations

While undertaking this comparative study, this project acknowledges a potential challenge arising from the inherent dependence of word embedding effectiveness on the specific characteristics of the dataset. Consequently, the model's findings may be more directly applicable to the nuances of the 20NewsGroups dataset. However, this limitation is intrinsic to NLP tasks, and this project aims to provide insights and recommendations that contribute meaningfully to the broader discourse on text classification. Insights derived from this analysis have the potential to inform best practices in text classification, guiding future endeavours in NLP and influencing the selection of word embeddings and neural architectures for optimal performance in diverse applications

3 Contributions

Table 1: Group members and their individual contributions.

Individual	**Contributions**	
Devynn Garrow	Implementation	GloVe pre-trained word embeddings, Dataset collection, Tokenization and Vectorization, Initial CNN, LSTM, + Combo Models
	Report + Presentation	Results and Discussion, Contributions

Jessica Guetre	Implementation	Dataset collection, Data preprocessing, Tokenization and Vectorization
	Report + Presentation	Related Work, Datasets, Implementation (Data Preprocessing)
Tom Hamilton	Implementation	Word2Vec pre-trained word embeddings, Dataset collection, Tokenization and Vectorization, Model Iteration
	Report + Presentation	Implementation (Experimental Setup, Training, Testing, and Validation)
Andy Craig	Implementation	Model Iteration, Modularization
	Report + Presentation	Motivation, Problem Description, Conclusions and Future Work

4 Related Work

4.1 Foundational Paper*

The first relevant paper reviewed in the literature review submission was "A Comparative Study on Word Embeddings in Deep Learning for Text Classification" [1]. The foundational paper illustrates the importance of word embedding selection, specifically by comparing classical and contextual word embeddings. Using the 20NewsGroups dataset, the study explores what neural network and embedding framework produces the most efficient model for text classification. A notable finding from the study is that classical word embeddings, including Word2Vec, GloVe and FastText, outperform contextualized embeddings like ELMo and BERT for long documents. This contradicts the general trend towards selecting more complex models. The implication is that simpler embeddings may be the more practical choice in analyzing long documents.

The paper "A Comparative Study on Word Embeddings in Deep Learning for Text Classification" implements a strong foundation for systematically comparing word embeddings. However, one weakness of the paper is the input length restrictions of contextual embeddings, causing a deficit in its ability to process long documents. The conclusion that no significant performance increase was observed for contextual embeddings on the 20NewsGroups dataset can be used in the context of this project to help interpret results. Further comparisons between the paper and this project models can be drawn from their shared use of Word2Vec and GloVe as classical word embeddings, CNN and LSTM as encoders, and the 20NewsGroups dataset. However, in contrast to the paper, which focuses on the comparative effectiveness of different embeddings, the model from this project addresses the combined result of different neural network architectures and hybrid models.

* Unless otherwise cited, content from each paper investigation comes from its corresponding paper

4.2 Relevant Journal Paper*

The model presented in "A unified approach to sentence segmentation of punctuated text in many languages," ERSATZ, centers on the task of sentence segmentation, a fundamental step in NLP pipelines [2]. Comparatively, this model focuses on classification, utilizing the segmented data for training and prediction. The strength of ERSATZ lies in its context-based design, which is adaptable to multiple languages and can be a crucial preprocessing step for this model's classification tasks.

The datasets used in the ERSATZ paper, primarily sourced from the WMT News Translation tasks, provide a multilingual basis for sentence segmentation. This diversity in data can enhance the robustness of the preprocessing steps in this model, mainly if multilingual classification is an objective. However, the paper acknowledges that it does not evaluate the impact of sentence segmentation on downstream tasks such as text classification, which this model directly addresses. The accuracy of ERSATZ is not directly comparable to this model as they serve different functions within the NLP pipeline. The model proposed by this project does not employ sentence segmentation and, therefore, cannot be compared. However, similarities between the models can be drawn from their shared use of preprocessing for the training and test data and a detailed display of results to conclude the model.

4.3 Paper Addressing Similar Issue*

"Hate Speech Detection Based on Sentiment Knowledge Sharing" proposes a model for hate speech detection using Sentiment Knowledge Sharing (SKS) [3]. The model's strength is its use of multi-task learning to improve hate speech detection by incorporating sentiment analysis. In comparison, this model may utilize sentiment analysis as an additional feature within its classification framework to potentially enhance. The datasets used in the SKS paper, such as the SemEval2019 task 5 and Davidson datasets, are curated explicitly for hate speech detection. This specificity contrasts with this model's use of the more general 20NewsGroups dataset, indicating a potential expansion area for this model to specialize in hate speech detection.

The SKS model reported improved accuracy and F1 scores over various baseline models, presenting a strong precedent for this model to measure against, especially if adapted for hate speech detection. The integration of sentiment analysis within the classification models could address a weakness in the SKS paper, where the imbalance in datasets led to potential overfitting. This project lacks similarity to the paper's model, given that it does not employ sentiment analysis. However, with sentiment analysis, this project serves as a framework for use in hate speech analysis, given its existing classification structure.

5 Datasets

The dataset used in training and evaluation is 20NewsGroups, a collection of around 20,000 newsgroup documents sectioned into 20 distinct newsgroups. 20NewsGroups is "a popular data set for experiments in text applications of machine learning," including "text classification and text clustering" [4]. Table 2 shows the categories of the dataset, partitioned by subject matter.

Table 2: 20NewsGroups categories partitioned by subject matter [5].

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

Each document in the dataset is given one of these labels to classify its subject matter. Table 3 shows a sample excerpt from the comp.graphics newsgroup, showing the type of text data the model was exposed to [4].

Table 3: Sample of the comp.graphics newsgroup from the 20NewsGroups dataset [4]

```
My brother is in the market for a high-performance video card that supports
VESA local bus with 1-2MB RAM.  Does anyone have suggestions/ideas on:

  - Diamond Stealth Pro Local Bus

  - Orchid Farenheit 1280

  - ATI Graphics Ultra Pro

  - Any other high-performance VLB card


Please post or email.  Thank you!

  - Matt
```

The numerous textual formats and linguistic styles can make it challenging for machine learning models to understand and categorize nuances in text effectively. However, this diverse data results in a model's enhanced ability to withstand various applications and scenarios

6 Implementation

6.1 Data Preprocessing

Starting with the unprocessed data from the 20NewsGroups dataset, such as that shown in Table 3, the data undergoes initial refining to remove any extraneous characters for more efficient training. This includes stripping special characters, removing whitespace, converting text to lowercase, and expanding contractions. Stopwords, common words such as "is" and "the" are also removed. Table 4 shows what the comp.graphics text data from above becomes after this process.

Table 4: Sample of the comp.graphics newsgroup from the 20NewsGroups dataset refined by the preprocessing code [4].

```
my brother market high performance video card supports vesa loc
al bus 1 2mb ram does anyone suggestions ideas diamond stealth
```

```
pro local bus orchid farenheit 1280 ati graphics ultra pro any
other high performance vlb card please post or email thank matt
```

Following the refining portion of preprocessing, the data is shuffled. The data is then tokenized, breaking it up into individual words or tokens. This is necessary to enable the model to process the text word-by-word. The final step is padding and vectorization. The padding is required to ensure that all the training and testing inputs are equal in length. Vectorization transforms the string tokens into integer values. In the case of pre-trained word embeddings like Word2Vec and GloVe, these values correspond to indices. The same comp.graphics text sample is shown in Table 5 after having undergone all of the vectorization process.

Table 5: Sample of the comp.graphics newsgroup from the 20NewsGroups dataset refined by the preprocessing code [4].

```
[    0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0    22  2160   988   272   769   522
   311  1283  2875   381   636     9  6044   846    71   122  1374   956
  2692  4030  1147   381   636  6664 72564 12499  2538   344  3357  1147
    38    55   272   769  2884   311   152   271     5   386  1059  2608]
```

Figure 1 shows a dataflow diagram depicting the entire preprocessing stage. Preprocessing is the same for all models analyzed.

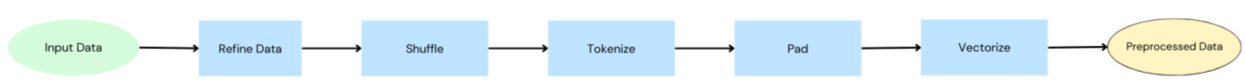

Figure 1: Dataflow diagram depicting the entire preprocessing stage.

6.2 Experimental Setup

The models were programmed in Python using Tensorflow and Keras. Additionally, NumPy and Pandas were used for preprocessing, Scikit-Learn was used to import and partition the dataset and to calculate performance metrics, and Matplotlib was used to generate the necessary plots for analysis. Jupyter Notebooks were used to separate parts of the model testing to improve

readability. All dependencies can be installed from the requirements.txt file using the 'pip install -r requirements.txt' command, as outlined in the read me file. Furthermore, the 300 features pretrained GloVe embeddings must be installed. This can be done using the GloVe 6B dataset on Kaggle [6]. Note that the Word2Vec pre-trained embeddings do not need to be preinstalled and are retrieved in the pretrained code using the GenSim library.

6.3 Model Architecture

Three models were developed to effectively compare different embedding and encoding techniques, utilizing distinct encoding methods for each. Subsequently, GloVe and Word2Vec were independently implemented into the three models, resulting in six variations.

The first model was a convolutional neural network (CNN), using convolutional layers to perform encoding. The goal was to have convolution and pooling layers to perform feature extraction on the texts after the pre-trained embeddings. Dropout layers were also included to prevent the model from overfitting and to increase robustness. The first layer of the model was an embedding layer with 300 output dimensions, followed by a dropout layer (50% rate). Next, a 1-D convolutional layer with 300 filters was followed by a max pooling layer, which was repeated once. A 1-D convolutional layer was then added, with the filters being reduced to 128, followed by another max pooling layer; this pair of layers was then repeated. Finally, a dropout layer was added (50% rate), followed by a dense layer using the SoftMax activation function with 20 output nodes to perform the classification. Note that a pool size of 3 was used for all max-pooling layers, and for all convolutional layers, ReLU activation was used. A diagram of the model can be seen in Figure 2.

The second model was a fully connected neural network that used LSTM. The intention for this model was to use LSTM to learn patterns in the text sequences and then use multiple fully connected layers to perform further feature extraction. Dropout was once again incorporated to help prevent overfitting. First, an embedding layer with 300 output dimensions, followed by a dropout layer (50% rate). Next, a dense layer with ReLU activation and 100 output nodes is followed by another dense layer using ReLU with 50 output nodes. The final layer was a dense layer using SoftMax activation with 20 output nodes to classify the data. A diagram of the model architecture can be seen in Figure 3.

The final model architecture used a combination of CNN and LSTM to perform encoding. By combining LSTM and convolutional techniques, the hope was for improved pattern recognition and feature extraction of the texts. The architecture was the same as the initial CNN model, with an additional LSTM layer before the final dense layer performing the classification. The model's architecture can be seen in Figure 4.

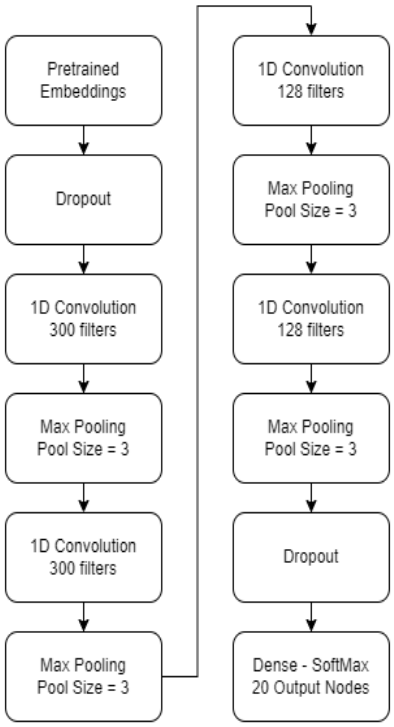

Figure 2: CNN model architecture.

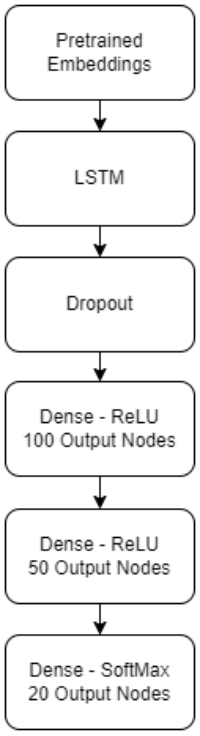

Figure 3: LSTM model architecture.

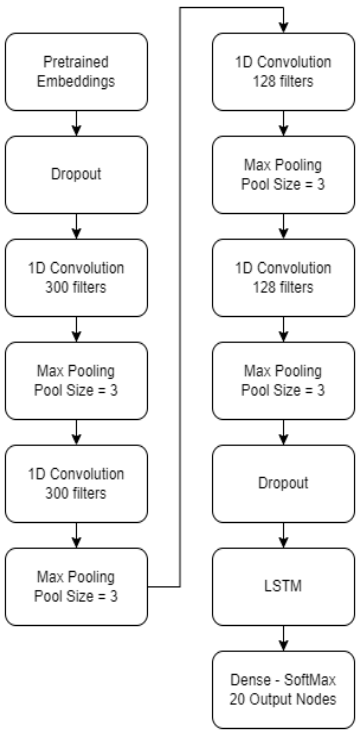

Figure 4: Architecture of CNN + LSTM model

6.4 Training, Testing, and Validation

Model training was a very iterative process, with model architectures and hyperparameters such as optimizers, number of epochs, batch size, etc., being tuned to find the ideal values. To fairly compare the different models, hyperparameters were kept the same across all models. The Adam optimizer, a variation of stochastic gradient descent that also considers momentum, was used in the final model. Normal stochastic gradient descent and Adagrad were also tested but produced worse results. The number of epochs for training was 30; this was determined by plotting the training and validation loss and observing when the model converged, and the loss stopped decreasing. Furthermore, the learning rate used was 0.001, 0.01 and 0.0001 were also experimented with, producing inferior results. As this is a classification task, sparse categorical cross entropy was used as the loss function.

The dataset was partitioned with 80% in the training and 20% in the testing sets. Further, 10% of the training set was further partitioned into the validation set to perform validation. To evaluate the models, the accuracy score was calculated on the training and validation sets during training and on the testing set after training was complete.

7 Results and Discussion

7.1 Results

Table 6: Results from implemented architectures and published models [1] to compare pre-trained word embeddings and classification models. Errors were obtained using results from 3 runs. Note that the last three columns were unavailable from the published models, thus are listed as N/A.

Source	Word Embedding	Model	Test Accuracy	Average Epoch Duration (s)	Epochs to 75% Validation Acc	Estimated time to 75% Validation Acc (mins)
Project Implementation	GloVe	CNN	79.74 ± 0.84 %	24.8 ± 0.5	~9	3.72 ± 0.08
		LSTM	76.50 ± 0.87 %	48.3 ± 2.0	~18	14.49 ± 0.60
		CNN + LSTM	79.64 ± 0.60%	26.1 ± 0.4	~11	4.79 ± 0.07
	Word2Vec	CNN	78.79 ± 0.91%	25.0 ± 0.3	~12	5.00 ± 0.06
		LSTM	73.25 ± 0.79%	37.8 ± 1.0	N/A	N/A
		CNN + LSTM	77.87 ± 0.64%	25.8 ± 0.6	~17	7.31 ± 0.17
Published Models	GloVe	CNN	82.61 ± 0.86%	N/A	N/A	N/A
		LSTM	74.24 ± 0.99%	N/A	N/A	N/A
	Word2Vec	CNN	82.67 ± 0.85%	N/A	N/A	N/A
		LSTM	62.45 ± 1.09%	N/A	N/A	N/A
	Baseline (Non Pre-trained)	CNN	78.00 ± 0.94%	N/A	N/A	N/A
		LSTM	49.02 ± 1.13%	N/A	N/A	N/A

Results are presented in Table 6, and are divided by word embedding, showing results for each model for each word embedding for the implementation described in Section 6.3, as well as the published models used in Wang, Nulty, and Lillis' study [1]. The architecture referred to as baseline uses the same classification models but does not use pre-trained embeddings. Instead, it uses a "randomly initialized trainable embedding layer" with an encoding vector dimension of 100 [1].

7.2 Discussion

The best architecture in test accuracy and computational cost is using GloVe pre-trained word embeddings with the CNN classification model, producing the highest test accuracy of all models, with results better than the published baseline model. However, this implementation is 2.87 ± 2.40% less accurate than the state-of-the-art model using the same architecture. The published architecture uses 100 filters of filter sizes 2, 3, and 4 with an Adam optimizer trained over 140 epochs with dropouts between 0.2 and 0.5 between layers [1]. Therefore, the source of this

improved accuracy is likely the extended training time, which was not used in this implementation due to computational limitations.

Comparatively, the architecture using GloVe with the LSTM classification model performs slightly worse in terms of accuracy than CNN; however, it also takes much longer to train, increasing computational cost and decreasing overall performance. This agrees with the expected results, as CNN works to extract features and reduce dimensionality, increasing training speed and decreasing computational cost. Although this architecture performs significantly worse than the implementation using CNN, it performs much better than the baseline LSTM model and slightly outperforms the published architecture using GloVe with LSTM. The published LSTM model uses a bidirectional LSTM with a hidden dimension of 256. In contrast, this implementation uses 100, indicating that this reduced hidden dimension size produces better results for natural language processing tasks, specifically when considering the 20NewsGroups dataset.

The final classification model used with GloVe embeddings combines CNN and LSTM, as described in Section 6.3. Although there is no comparable model in the published architecture considered, this architecture performs similarly to the implementation using CNN alone with respect to model accuracy and training time. Therefore, to differentiate the performance between this model and CNN alone, the number of epochs to reach a 75% validation accuracy can be considered, indicating that CNN alone reaches this threshold earlier, leading to the conclusion that the architecture using solely a CNN classification model is the best solution when using GloVe pre-trained word embeddings. Figure 5 summarizes these results between models for the GloVe pre-trained word embeddings.

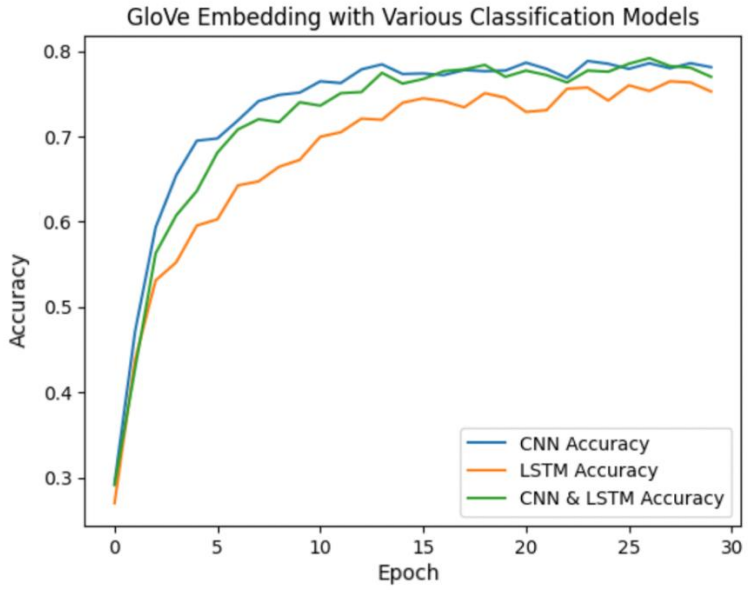

Figure 5: Classification Model Comparison of Validation Accuracy for GloVe Pre-trained Word Embeddings

Similar model-specific trends can be observed for architectures using Word2Vec pre-trained embeddings and can be compared to GloVe embeddings to evaluate their relative performances. As expected, for Word2Vec, the implementation using a CNN classification model performs the best in terms of test accuracy and computational duration. This implementation also performs

better than the baseline CNN architecture; however, these two architectures perform similarly when accounting for error. This performance reduction, compared to a similar implementation with GloVe, can also be observed in the decrease in accuracy between this implementation and the published model, at 4.79 ± 1.20% %, which is more than this comparison for GloVe.

The architecture using Word2Vec with LSTM also produces lower accuracy than the implementation using GloVe, but this implementation still vastly outperforms the published baseline model and quite significantly outperforms the published architecture using Word2Vec with LSTM. This supports the prior conclusion that the LSTM model employed in this implementation is superior to that described in relevant publications, especially when using Word2Vec embeddings. As expected, classification using this LSTM model results in much longer training times than those using CNN; however, it has a noticeably shorter average epoch duration than the same classification model with GloVe embeddings.

Finally, the combination model using Word2Vec embeddings performs similarly to the model using CNN alone regarding test accuracy and computation time; however, it produces a slightly lower accuracy than this architecture with GloVe embeddings. Additionally, although the average epoch durations are similar for this classification model with both embeddings, the architecture with Word2Vec takes much longer to reach a 75% validation accuracy than the comparable architecture with GloVe. The Word2Vec embedding-specific results are compared in Figure 6, and lead to the same conclusions of optimal performance using the CNN model.

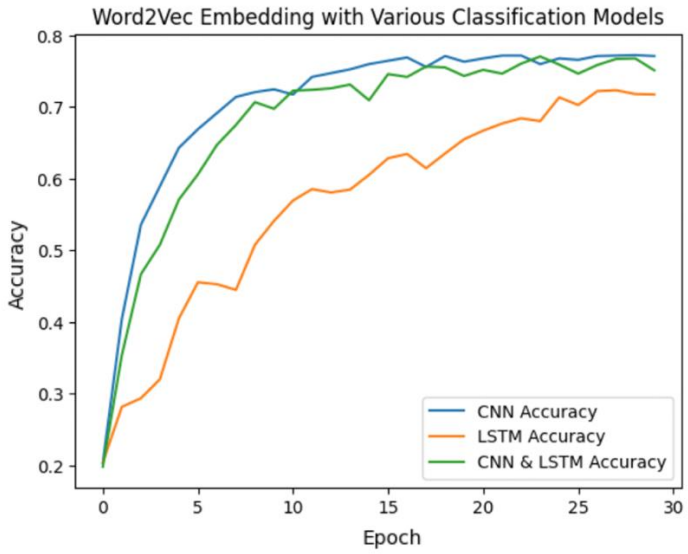

Figure 6: Classification Model Comparison of Validation Accuracy for Word2Vec Pre-trained Word Embeddings

A summary of these comparisons are displayed in Figure 7 and Figure 8. Figure 7 describes the embedding-independent comparison, averaging results across both embeddings for all classification models. As expected, the architecture using a CNN classification model outperforms the others, regardless of embedding choice, as it performs better for both embeddings individually. Additionally, LSTM alone produces significantly lower accuracies throughout the entire training period, which was observed for all architectures.

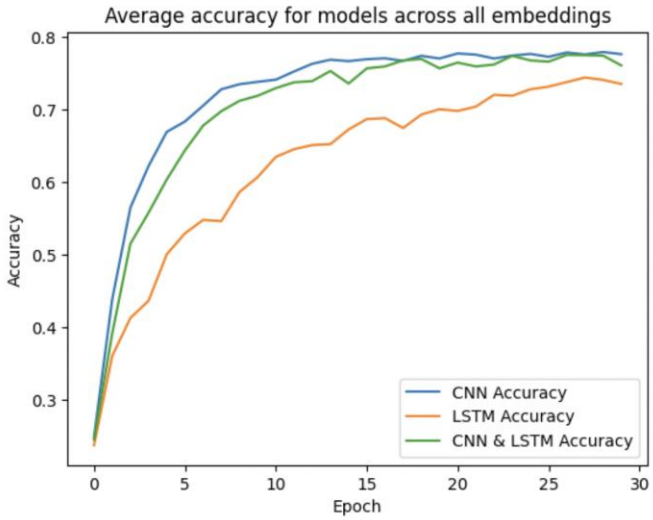

Figure 7: Embedding-independent Classification Model Comparison of Validation Accuracy

Figure 8 includes a model-independent comparison, supporting the conclusion that architectures using GloVe word embeddings perform better than those using Word2Vec embeddings, specifically in test accuracy.

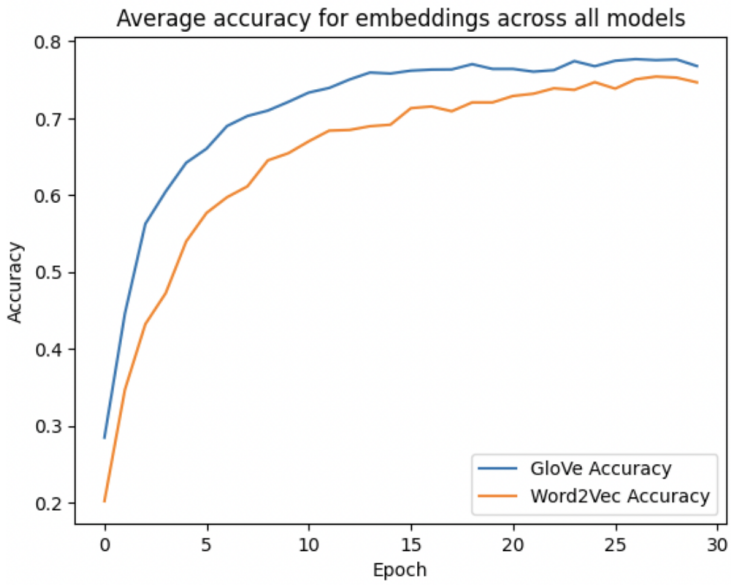

Figure 8: Model-independent Embedding Comparison of Validation Accuracy

These comparisons lead to the recommendation that for Natural Language Processing classification tasks, specifically using 20NewsGroup as the training set, the best performance is produced by architectures using GloVe word embeddings with a CNN classification model. Similar results are obtained for classification models consisting of a combination of CNN and LSTM; however, there is no observed benefit in including an LSTM layer in the model. This is the same conclusion that has been drawn in published comparisons, specifically in Wang, Nulty, and Lillis' study [1].

8 Conclusion and Future Work

In summary, this project focuses on implementing and comparing three distinct models: CNN, LSTM, and a combination of CNN with LSTM, each utilizing GloVe and Word2Vec embeddings. The primary objective was to assess their efficacy in text classification using the 20NewsGroups dataset. Among these models, the CNN architecture paired with GloVe embeddings consistently outperformed others, yielding the highest accuracy.

While the project may not revolutionize the broader field of NLP, the findings highlight the significance of specific embedding-model combinations. The success of GloVe with CNN suggests the importance of considering both embedding type and neural architecture for optimal results in text classification tasks.

The standout performer was the model employing GloVe embeddings with the CNN architecture, achieving a test accuracy of 79.74% with an average epoch duration of 24.8 seconds. This combination demonstrated superior performance compared to other configurations, including Word2Vec embeddings and LSTM architectures.

8.1 Areas for Improvement:

Despite the promising outcomes, certain challenges remained unaddressed. Notably, the effectiveness of word embeddings is intricately tied to the dataset's characteristics. This raises a potential limitation, indicating that these findings might be more specific to the nuances of the 20NewsGroups dataset. A broader evaluation across diverse datasets could provide a more comprehensive understanding of the generalizability of the results. Additionally, the following areas could be explored in future work:

1. *Different Datasets:* To enhance the robustness of the conclusions, an extended evaluation on different datasets could help validate the effectiveness of the identified optimal configuration across varied text corpora.
2. *Hyperparameters:* Fine-tuning hyperparameters, such as dropout and learning rates, was beyond this project's scope. A more exhaustive exploration in this regard might yield further improvements in model performance.
3. *Combination Models:* Exploring combinations of models by integrating aspects from multiple models or embeddings could be explored to potentially boost overall classification accuracy.
4. *Other NLP Tasks:* While this project focused on text classification, extending the models to other NLP tasks, and assessing their adaptability would contribute to a more comprehensive understanding of their utility.

In conclusion, this project lays the groundwork for future endeavours by identifying optimal configurations for text classification tasks. Acknowledging the challenges and proposing areas for improvement ensures that the findings of this model serve as a stepping stone for further exploration and refinement in the dynamic field of NLP.

9 Work Cited

[1] C. Wang, P. Nulty, and D. Lillis, "A Comparative Study on Word Embeddings in Deep Learning for Text Classification," Dec. 2020, pp. 37–46. doi: 10.1145/3443279.3443304.

[2] "A unified approach to sentence segmentation of punctuated text in many languages." Accessed: Nov. 29, 2023. [Online]. Available: https://paperswithcode.com/paper/a-unified-approach-to-sentence-segmentation

[3] X. Zhou *et al.*, "Hate Speech Detection Based on Sentiment Knowledge Sharing," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds., Online: Association for Computational Linguistics, Aug. 2021, pp. 7158–7166. doi: 10.18653/v1/2021.acl-long.556.

[4] "20 Newsgroups." Accessed: Dec. 02, 2023. [Online]. Available: https://www.kaggle.com/datasets/crawford/20-newsgroups

[5] "Home Page for 20 Newsgroups Data Set." Accessed: Dec. 02, 2023. [Online]. Available: http://qwone.com/~jason/20Newsgroups/

[6] "GloVe 6B." Accessed: Dec. 04, 2023. [Online]. Available: https://www.kaggle.com/datasets/anindya2906/glove6b