Leveraging Convolutional Neural Networks for Accurate News Article Classification: A Study on the AG News Dataset

Agron Gojcaj & Kyle Verellen
Oakland University – CSI 4180
Auburn Hills, Michigan
agrongojcaj@oakland.edu, kverellen@oakland.edu

Abstract– The exponential growth of digital news content necessitates efficient and accurate classification systems to organize and analyze textual information. This study explores the application of Convolutional Neural Networks (CNNs) for news article classification using the AG News Dataset, which categorizes articles into domains such as sports, politics, technology, and more. The methodology involves preprocessing the text data through cleaning, punctuation handling, and lemmatization. Word embeddings are employed to capture the semantic and contextual relationships within the text. The CNN model leverages these embeddings to learn hierarchical features, enabling robust classification. Evaluation metrics, including accuracy and F1-score, demonstrate the effectiveness of CNNs in capturing complex textual patterns compared to traditional methods. This research highlights the potential of deep learning in handling large-scale text classification tasks, offering a scalable and efficient solution for organizing news content in real-world applications.

I. Introduction

The explosive growth of online news content has created a pressing need for tools that can efficiently categorize and organize textual data. Traditionally, news classification has relied on manual effort, where articles are sorted into categories like politics, sports, and technology. This approach is time-intensive and impractical with the sheer volume of modern digital content. The challenge lies in developing automated systems capable of handling this task accurately and efficiently.

In recent years, advances in natural language processing (NLP) have provided new opportunities for tackling text classification. Techniques like machine learning and deep learning, particularly Convolutional Neural Networks (CNNs), have shown promise in identifying patterns within textual data. CNNs, originally designed for image processing, have been adapted to handle sequential data by capturing relationships between words and phrases. Researchers have explored models that incorporate word embeddings, attention mechanisms, and dynamic pooling layers to enhance performance in tasks such as sentiment analysis, spam detection, and text categorization.

This project builds on these advancements by applying CNNs to the AG News Dataset, a collection of articles grouped into four categories: world news, sports, business, and sci/tech. By preprocessing the text through techniques like noise removal and lemmatization and leveraging word embeddings, the goal is to design a model that learns nuanced patterns for effective classification. This work aims to achieve similar results to those observed in prior research, where CNNs were used to address text classification challenges effectively. Specifically, the goal is to replicate their success in balancing model complexity and performance, focusing on capturing relationships between words and categories for improved classification accuracy. By benchmarking a different approach against similar models, the aim is to validate its performance through metrics like accuracy and F1-score, ensuring competitive results with state-of-the-art methods.

By addressing the challenges of text classification, this research not only aims to create a practical solution for news categorization but also contributes to the broader development of NLP tools that can be applied in real-world scenarios, from content management to personalized recommendations.

II. Data Handling

To handle the data of the AG News Dataset from Kaggle, the first technique used was data cleaning. Data cleaning was used to convert all the text data to lowercase letters, and also to remove punctuation and any special characters. Converting the text to lowercase was done to ensure consistency with tokenization and to reduce the size of the vocabulary of tokens. Removing punctuation and special characters were done to reduce noise, as they do not carry semantic meaning for our classification task.

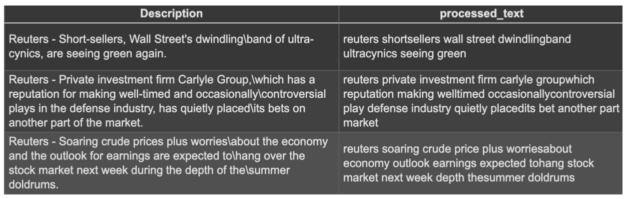
The next technique used was stopword removal using the Natural Language Toolkit (nltk). A stopword is a frequently used word that is considered to have little meaning on its own and doesn’t contribute significantly to the overall context. For example, ‘is’, ‘and’, and ‘an’ are examples of stopwords. The stopwords were removed to reduce the noise of our data.

Another preprocessing technique used was lemmatization. Lemmatization is the process of reducing words into their base form, or lemma, using a vocabulary or a part-of-speech tagging. This is done to reduce noise, overfitting, and to greatly reduce the vocabulary size, as it treats inflected forms of words the same way.

For our Naive Bayes model, data was preprocessed with feature extraction. Feature extraction is used to convert text into numerical data so it is able to be input into the model. In this case, TF-IDF (Term Frequency-Inverse Document Frequency) was utilized using TfidfVectorizer in the scikit-learn library. It transformed our text data into a sparse matrix of numerical features. The importance of feature extraction with TF-IDF is to emphasize the uncommon, but important words that may be unique to a category with a higher weight, while also assigning a lower weight to words that are more common.

For our CNN model, tokenization and padding were necessary to prepare for the neural network. Tokenization is defined as the process of breaking down a piece of text into smaller units called tokens, which are typically words or characters, allowing a machine to better understand the text by dividing it into more manageable parts. In this problem, text data was converted into sequences of integers using the Tokenizer in the Keras library. Those sequences were then padded to a fixed length using the pad_sequences function from Keras. These steps are necessary to prepare for the embedding layer of the CNN, which allows it to learn semantic relationships between words. It also ensures that the training is efficient since the sequences are all the same length.

See **Figure 1** for examples of the preprocessed text.

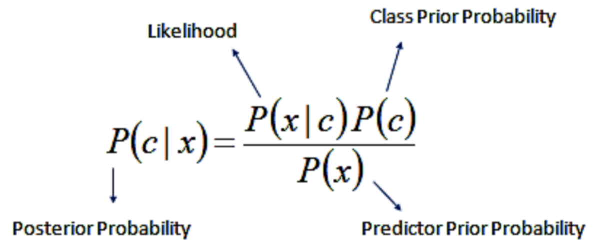


Description	processed_text
Reuters - Short-sellers, Wall Street's dwindling\band of ultra-cynics, are seeing green again.	reuters shortsellers wall street dwindlingband ultracynics seeing green
Reuters - Private investment firm Carlyle Group,\which has a reputation for making well-timed and occasionally\controversial plays in the defense industry, has quietly placed\its bets on another part of the market.	reuters private investment firm carlyle groupwhich reputation making welltimed occasionallycontroversial play defense industry quietly placedits bet another part market
Reuters - Soaring crude prices plus worries\about the economy and the outlook for earnings are expected to\hang over the stock market next week during the depth of the\summer doldrums.	reuters soaring crude price plus worriesabout economy outlook earnings expected tohang stock market next week depth thesummer doldrums

Figure 1

III. Model Selection and Implementation

The first model chosen and implemented for our text classification project was a Multinomial Naive Bayes model. The purpose of this model is to provide a simple baseline model to compare against our more complex CNN model. It is used to determine if the use of our CNN model is justified. The Naive Bayes algorithm is a probabilistic model based on Bayes' Theorem, shown in **Figure 2.** The equation calculates the posterior probability by multiplying the likelihood by the class prior probability, and then dividing the product by the predictor's prior probability.



$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

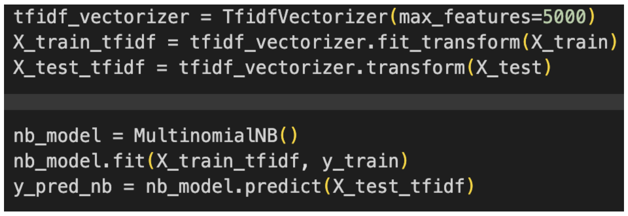
$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Figure 2

Naive Bayes is computationally 'cheap' and quick to train, even on large datasets like the AG News dataset. When coupled with TF-IDF feature extraction, it can be effectively used for text classification.

A Multinomial Naive Bayes model works for this project because text classification problems often rely on the frequency and importance of specific words. The fact that Naive Bayes assumes conditional independence, although it is rarely true, works well for this problem because many words contribute independently to the class probability.

The model was implemented using scikit-learn's MultiNomialNB() function, where train_test_split was used to split the dataset into 80% in the training set and 20% in the testing set. See **Figure 3** for the implementation of the model using TF-IDF feature extraction.



```
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

nb_model = MultinomialNB()
nb_model.fit(X_train_tfidf, y_train)
y_pred_nb = nb_model.predict(X_test_tfidf)
```

Figure 3

The other model that was chosen and implemented was a Convolutional Neural Network (CNN) using the Keras library. A CNN is a deep learning model designed to automatically extract patterns from data. Although it was initially developed for image processing, CNNs are also highly effective on text data for tasks such as text classification and sentiment analysis. The network begins with the input layer, where raw data is received and converted into a suitable form for processing. In this layer, tokenization and word embedding is done to ensure the data is in a suitable format for processing. In the next layer, the convolution layer, filters slide over the embeddings and patterns are extracted, such as n-grams or word sequences. Non-linear functions such as ReLU are applied to the results of the convolution, which enable the model to learn complex relationships. In the next layer, the pooling layer, the spatial dimensions of input data are reduced while retaining the most important information. The fully connected layers combine the extracted features to then make a prediction. Finally, in the output layer, a softmax layer outputs the probabilities for each class. A visual representation of a CNN is shown in **Figure 4.**

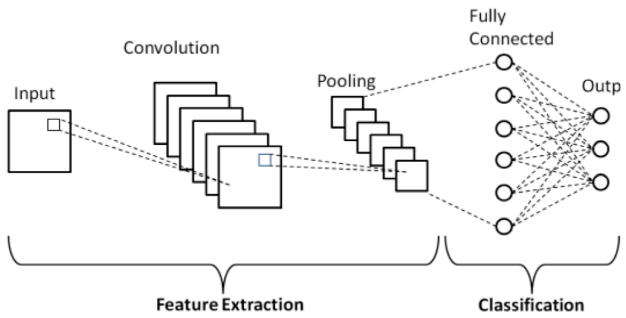


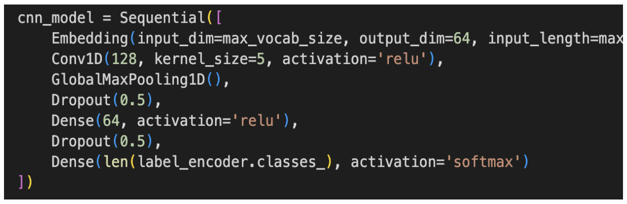
Figure 4

A CNN was selected because they excel at identifying local patterns in text, which is crucial for our text classification problem. For example, a phrase like “soccer match” is a good indicator that the article is labeled in the sports category. Another reason for its selection is that it processes with fixed-length sequences (through padding), meaning that the model focuses on the most important patterns, regardless of input size. We wanted a model that can capture semantic relationships between words, which a CNN does through dense word embeddings. CNNs are also proven to perform strongly for text classification problems, as well as similar problems such as sentiment analysis, topic classification, and spam detection.

The reason that a CNN works for this project is because it captures local context very effectively, it performs feature extraction with convolution, it is robust to irrelevant information (through max pooling), and it is compatible with word embeddings. With the dataset being quite large, there is enough data for a CNN to learn effectively.

Since the articles in the AG News dataset are relatively short, a Recurrent Neural Network (RNN) would not be as efficient and would be much more computationally expensive compared to a CNN. An RNN processes input sequentially, or one word at a time, whereas a CNN processes data in parallel, thus making it more efficient and faster to train and predict. Also, for our problem, we only need to capture local context, which a CNN excels in. An RNN would be better suited to a problem that requires understanding of long-term dependencies of the entire sequence, such as text generation or language translation.

The implementation of our CNN model is shown in **Figure 5**. We begin by initializing our Sequential model, which allows us to stack our layers sequentially. In the following line, each word in the text is converted into a Dense vector with a size of 64 elements. Next, a 1-dimensional convolutional layer is added with 128 filters and a kernel size of 5. This learns word combinations and phrases in the text. Then, max pooling is performed across the output of the convolutional layer to reduce data size while also preserving important features. In the next layer, Dropout(0.5) randomly sets 50% of the weights to zero during training, preventing overfitting. After that, a fully connected dense layer is formed with 64 neurons and the Rectified Linear Unit (ReLU) activation function. The ReLU function works by mapping an input value to the maximum of either the input or zero. If the input is positive, then the output is the same as the input. If the input is negative, then the output is zero. ReLU is used to introduce non-linearity and help the neural network learn complex patterns in data. Following this layer, another dropout layer is added to further regularize the model and reduce overfitting. The final layer of the CNN model is the output layer, where a softmax activation function is used to convert the raw output scores into probabilities for each class, where all of the probabilities add up to equal 1.



```
cnn_model = Sequential([
    Embedding(input_dim=max_vocab_size, output_dim=64, input_length=max
    Conv1D(128, kernel_size=5, activation='relu'),
    GlobalMaxPooling1D(),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(len(label_encoder.classes_), activation='softmax')
])
```

Figure 5

IV. Results

To evaluate the performance of the Naive Bayes model as well as the CNN model, accuracy score, F1-score, and confusion matrices from the scikit-learn library were implemented. In addition, the matplotlib library as well as the seaborn library were used to create visual representations of the evaluation metrics.

Accuracy score is a metric that measures how often a model correctly predicts the outcome. The formula for accuracy score is the number of correct predictions divided by the total number of predictions. It was used to evaluate our models because it provides a general and reliable indication of how the model performs. For our Naive Bayes model, an accuracy score of 88.42% was achieved. For our CNN model, an accuracy score of 89.68% was achieved, slightly improving on the Naive Bayes model. It fell slightly short of the Text-CNN model that we found while researching this topic, which had an accuracy of 92.46%.

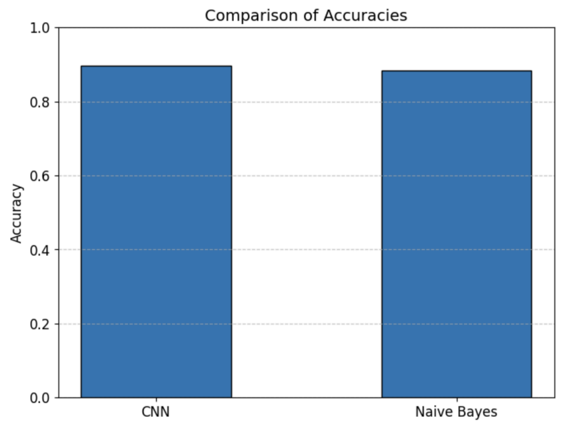


Figure 6

F1-score is a metric that measures the harmonic mean between precision and recall, with precision measuring how many predicted true positives were actually correct and recall measuring how many positive cases were correctly predicted. See **Figure 7** for the formula used to calculate F1-score.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Figure 7

The F1-score for the Multinomial Naive Bayes model was 88.38%. The CNN model achieved an 89.67% F1-score, once again improving slightly on the Naive Bayes model.

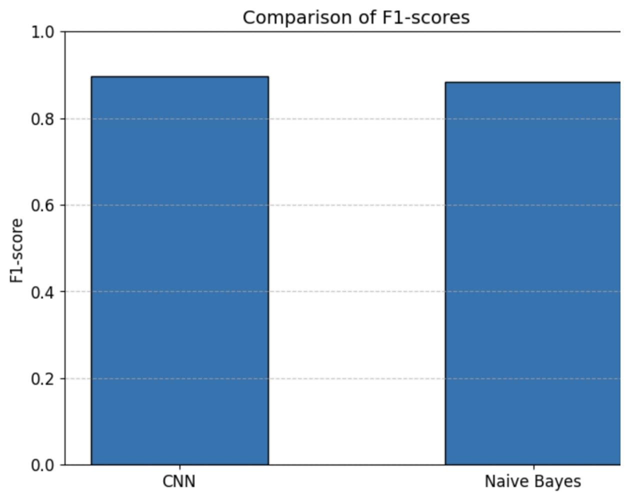
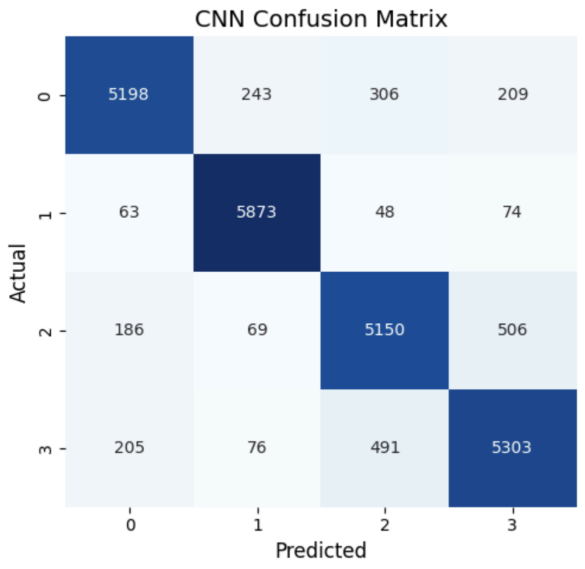


Figure 7

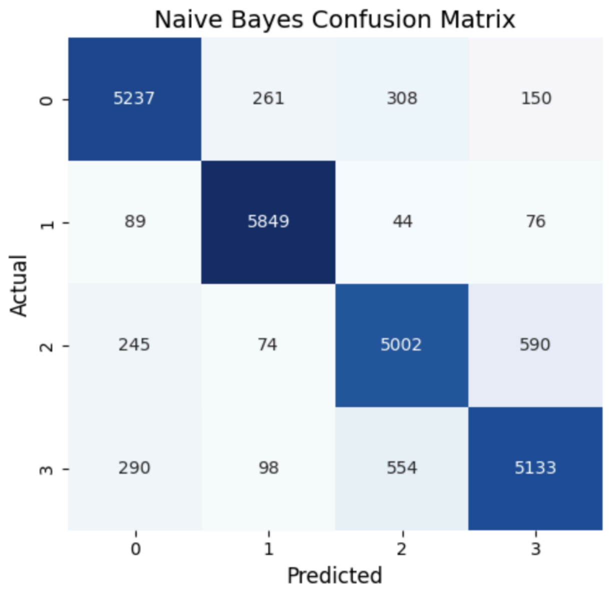
The final metric used to evaluate the models was a confusion matrix. A confusion matrix displays the count of correctly predicted positive class instances (true positives), correctly predicted negative class instances (true negatives), incorrectly predicted positive instances (false positives), and missed positive instances (false negatives).

Confusion matrices were used to reveal exactly where the model is making mispredictions, making the model easily interpretable. It also helps to identify which classes are being misclassified more frequently. For example, in both the Naive Bayes and the CNN models, the "World News" articles were misclassified the most, indicating that the models need improvement in predicting this class. "Business" and "Sci/Tech" texts also had quite a bit of mispredictions, while the "Sports" texts were most accurately predicted. See the CNN confusion matrix in **Figure 8** and the Naive Bayes confusion matrix in **Figure 9**.



Actual \ Predicted	0	1	2	3
0	5198	243	306	209
1	63	5873	48	74
2	186	69	5150	506
3	205	76	491	5303

Figure 8



Actual \ Predicted	0	1	2	3
0	5237	261	308	150
1	89	5849	44	76
2	245	74	5002	590
3	290	98	554	5133

Figure 9

V. Conclusion

In this project, two models—a Multinomial Naive Bayes (MNB) and a Convolutional Neural Network (CNN)—were implemented to classify news articles from the AG News Dataset. The preprocessing steps, such as stopword removal, lemmatization, and tokenization, ensured clean and structured data for effective model training.

The MNB model achieved an accuracy of 88.42% and an F1-score of 88.38%, serving as a robust baseline for comparison. The CNN model improved upon these metrics, reaching an accuracy of 89.68% and an F1-score of 89.67%, demonstrating its capability to capture semantic relationships in the text. However, the CNN model fell slightly short of the 92.46% accuracy reported by the Text-CNN model in prior research.

While the CNN model demonstrated strong classification performance, particularly in the "Sports" category, challenges persisted in accurately predicting "World News" and other categories, as revealed in the confusion matrix analysis. This highlights the potential for further optimization.

By comparing these findings to the results of prior research, it is evident that while this CNN implementation approaches the performance of state-of-the-art models, enhancements in preprocessing (e.g., using specialized stopword lists) and architectural refinement (e.g., integrating attention mechanisms) could further boost accuracy and robustness. Future work could also address dataset imbalances and explore hybrid models to close the gap with cutting-edge approaches.

VI. References

[1] A. Anand, “AG News Classification Dataset,” *www.kaggle.com*, 2020. https://www.kaggle.com/datasets/amananandrai/ag-news-classification-dataset

[2] geeksforgeeks, “Convolutional Neural Network (CNN) in Machine Learning,” *GeeksforGeeks*, Dec. 25, 2020. https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/

[3] M. Zhang, “Applications of Deep Learning in News Text Classification,” *Scientific Programming*, vol. 2021, pp. 1–9, Aug. 2021, doi: https://doi.org/10.1155/2021/6095354.

[4] Y. Zhu, “Research on News Text Classification Based on Deep Learning Convolutional Neural Network,” *Wireless Communications and Mobile Computing*, vol. 2021, pp. 1–6, Dec. 2021, doi: https://doi.org/10.1155/2021/1508150.

[5] “CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization,” *ieeexplore.ieee.org*. https://ieeexplore.ieee.org/abstract/document/9222325

[6] S. Xu, Y. Li, and Z. Wang, “Bayesian Multinomial Naïve Bayes Classifier to Text Classification,” *Lecture Notes in Electrical Engineering*, pp. 347–352, 2017, doi: https://doi.org/10.1007/978-981-10-5041-1_57.