Deep Learning for Arabic Sentiment Analysis

MOHAMMED V UNIVERSITY

Faculty of Science Rabat

TOUIJER OUSSAMA
Master (IAO)
Faculty of Science Rabat
Rabat, Morocco
oussama.touijer@um5r.ac.ma

ASSAL ANASS
Master (IAO)
Faculty of Science Rabat
Rabat, Morocco
anass.assal@um5r.ac.ma

***Abstract*—Sentiment analysis is a crucial task in natural language processing, with applications ranging from market analysis to social media monitoring. This paper explores the use of Convolutional Neural Networks (CNNs) for Arabic sentiment analysis, a challenging task due to the complex morphology and rich morphology of the Arabic language.**

We propose a CNN-based model that leverages word embeddings and various preprocessing techniques to classify Arabic text into positive and negative sentiments. Our experiments on a dataset of 60,000 Arabic reviews demonstrate the effectiveness of the proposed model, achieving an accuracy of 84%.

The results suggest that CNNs are a promising approach for Arabic sentiment analysis, though further improvements can be made with more specialized datasets. This work contributes to the growing body of research on Arabic NLP and provides a foundation for future studies in this area.

***Index Terms*—Arabic Sentiment Analysis, Deep Learning, Convolutional Neural Networks, Natural Language Processing, Word Embeddings,**

I. Introduction

The main objective of this project is to use a Deep Learning model, specifically a Convolutional Neural Network (CNN), to perform sentiment analysis on an Arabic dataset. This approach aims to detect and classify sentiments (positive and negative) expressed in Arabic texts, which is crucial for applications such as review analysis, content moderation, or studying opinion trends.

The CNN model is chosen because of its ability to capture local patterns in text data, which is particularly useful for processing text sequences. CNN can be very effective in extracting relevant information from texts for the development of this model.

This document was written as part of a final project for the BI DM module. At the end of this work, we would like to express our deep gratitude and sincere thanks to our supervisor, Professor **BEN KHALIFA Mohammed**.

Kaggle was used to store the dataset and facilitate initial data exploration, leveraging its built-in tools for visualization and preprocessing . The implementation and training of the deep learning model were carried out on Google Colab, which provides a cloud-based development environment with hardware resources such as GPUs,enabling faster and more efficient model training. Additionally, Colab simplifies the integration of popular Python libraries, making it an ideal platform for building and optimizing neural networks.

II. Convolutional Neural Network (CNN)

The CNN (Convolutional Neural Network) model is commonly used in image processing tasks, but it can also be effective for text processing. In sentiment analysis, the idea is to treat text as a sequence of data where each word or group of words can be represented as dense vectors (embeddings) that capture the semantic relationships between words. The CNN can then detect local patterns in these text sequences to extract relevant features for sentiment classification purposes.

A. CNN layers

CNNs for text analysis typically consist of several layers:

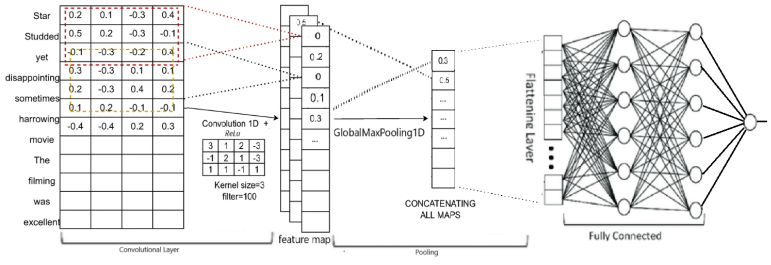

Figure 1. Architecture of CNN

There are two main layers in the structure of the CNN algorithm: convolution layer and pooling layer. In the sentiment analysis, we simply use the two main layers illustrated in Figure 1.

1) convolutionl layer: In a CNN, the convolutional layer plays a key role in extracting local features from input data. It applies filters that move across the input matrix, performing a dot product with corresponding regions. Through gradient descent and backpropagation, the filter and bias parameters are optimized automatically. This layer identifies important patterns, such as word relationships in text, and produces feature maps, which are then processed by pooling and fully connected layers for classification or other NLP tasks.

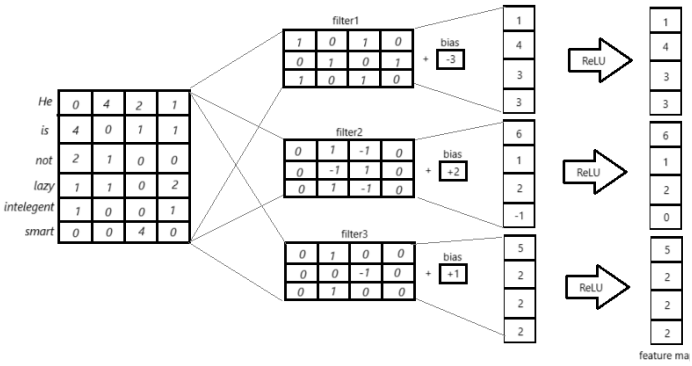

Figure 2. Convolutional layer

2) Pooling Layer: The pooling layer in a CNN reduces the dimensionality of feature maps while preserving key information, lowering model complexity and minimizing overfitting. MaxPooling selects the maximum value in each region, emphasizing important features, while AveragePooling computes the average, creating a smoother representation. GlobalMaxPooling differs by taking the maximum value from the entire feature map, significantly reducing dimensions and simplifying the model. It converts feature maps into a single value, making it ideal for classification tasks.

GlobalMaxPooling aligns with CNNs as it preserves the most dominant features detected by convolutional layers while maintaining spatial invariance. Its advantages include fewer parameters, reduced computation cost, better generalization, and improved robustness to input variations. However, its main drawbacks are the loss of spatial information, which may reduce model performance in tasks requiring fine-grained details, and its sensitivity to outliers, as a single high activation can dominate the output, potentially leading to misleading representations.

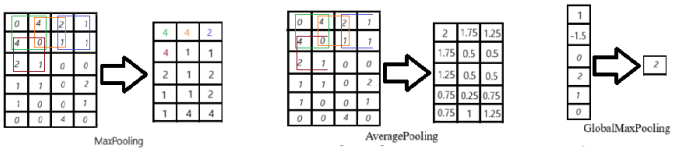

Figure 3. Kinds of Pooling

3) Fully Connected Layer: In Sentiment Analysis using Convolutional Neural Networks (CNNs), the Fully Connected (FC) Layer plays a crucial role in making final predictions. Typically placed at the end of the network, this layer connects every neuron to all neurons in the previous layer, integrating the high-level features extracted by convolutional and pooling layers.

During training, the FC layer learns to weigh these features appropriately by adjusting its weights and biases through backpropagation. In forward propagation, each neuron applies a linear transformation (weight multiplication and bias addition) followed by an activation function to introduce non-linearity. The network then calculates the error between its prediction and the actual sentiment label, updating the weights iteratively to minimize this error.

By transforming extracted features into a final decision, the FC layer ensures that CNNs can effectively classify sentiments, making it an essential component in deep learning models for text analysis.

III. Our Proposed Model

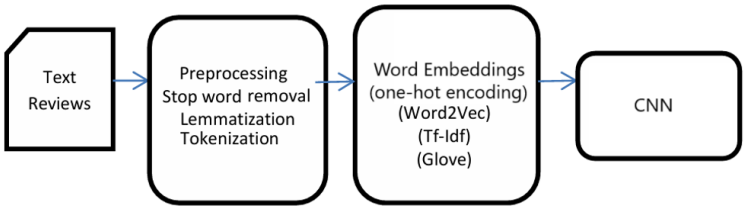

Figure 4. Diagram of the Proposed model

A. Pre-processing Techniques

1) cleaning: Text cleaning is a fundamental step in the pre-processing phase of sentiment analysis, ensuring that the input data is well-structured and free from unnecessary noise. This process involves removing special characters, punctuation, and numerical digits that do not contribute to the semantic meaning of the text. Additionally, extra whitespaces are eliminated to maintain a standardized text format. By applying these techniques, the dataset becomes more refined, allowing the model to focus on meaningful linguistic patterns without interference from irrelevant element.

2) Tokenization: Tokenization breaks text into smaller units (tokens), typically words or subwords, allowing the model to process them effectively. In Arabic, it helps manage complex morphology by separating prefixes and suffixes. Removing unnecessary tokens enhances sentiment analysis accuracy.

3) Stop word Removal: There are certain words such as conjunction and preposition that are considered irrelevant words. These words or features are known as stop words. Those words provide no meaning to the text. The process of eliminating those words is known as Stop word removal.

4) Lemmatization: The process of transforming the word to its base word that is uniquely identified in the IR dictionary is known as Lemmatization. Typical text reviews are high-dimensional. Lemmatization is widely used to reduce the number of dimensions in text reviews.

B. Word Embeddings

Word embeddings are a fundamental technique in Natural Language Processing (NLP) that represent words as vectors in a continuous space. Unlike traditional discrete representations such as one-hot encoding, embeddings capture semantic relationships between words by learning their usage contexts from large text corpora. This approach enables machine learning models to better understand word similarities and improve performance in various tasks, including text classification, machine translation, and sentiment analysis. Several methods exist, including pre-trained embeddings like Word2Vec and GloVe, as well as embeddings trained from scratch, which are tailored to specific corpora.

IV. Hyperparameter Tuning and Optimization Methods

A. Filter Size

- The filter size determines the number of words considered in each convolutional operation.
- A filter size of 4 captures local n-gram features (such as phrases or word combinations of 4 words).
- Increasing the filter size could help detect longer patterns but may reduce sensitivity to short but meaningful phrases.

B. Number of Filters

- Defines the number of convolutional kernels applied to the text sequences.
- More filters allow the model to learn diverse patterns from text, improving feature extraction.
- However, a higher number of filters increases computational complexity and training time.

C. Embedding Size

- This represents the dimensionality of word embeddings, transforming words into dense vectors.
- A size of 50 balances capturing semantic relationships while keeping computational cost reasonable.
- Larger embeddings can store richer word meanings but require more training data to be effective.

D. Learning Rate

- Determines the step size for weight updates during training.
- A low learning rate (0.0001) ensures stable convergence and prevents overshooting optimal weights.
- A higher learning rate could speed up training but might cause instability.

E. Batch Size

- Defines how many samples are processed before updating model weights.
- A batch size of 128 ensures stable training, reducing variance in gradient updates.
- Larger batch sizes may improve efficiency but require more memory.

F. Number of Epochs

- The model is trained for 8 full passes over the dataset.
- This number balances learning efficiency with the risk of overfitting.
- More epochs may be needed if the model hasn't fully converged, while fewer epochs might lead to underfitting.

To achieve better results, hyperparameters should be adjusted based on model performance and dataset characteristics. Filter size can be increased (e.g., from 4 to 5 or 6) to capture longer contextual patterns, while the number of filters can be tuned (128 to 256) to enhance feature extraction. If the model underperforms, embedding size may be increased (50 to 100) to provide richer word representations. Learning rate should be carefully adjusted; if training is too slow, increasing it slightly (0.0001 to 0.0005) can help, but too high a value may lead to instability. If overfitting occurs, reducing batch size (128 to 64) or applying regularization techniques like dropout can help. Additionally, increasing epochs (8 to 15) can allow the model to learn better, provided there is no overfitting. Fine-tuning these hyperparameters using validation performance and loss curves is essential for achieving optimal results.

V. Using Google Colab

Google Colab provides a cloud-based environment for training deep learning models without requiring high-end local hardware. One of its strengths is the availability of free access to powerful GPUs, which significantly accelerates training, especially for resource-intensive tasks like deep learning. Additionally, Colab offers seamless integration with Google Drive, making it easier to manage datasets and save model checkpoints. With pre-installed libraries like TensorFlow, Keras, and PyTorch, it simplifies the setup process, and users can start training models quickly without worrying about compatibility issues. Below are the steps to set up the workspace and prepare the dataset for training a CNN model for Arabic sentiment analysis.

Installation of Required Libraries: For this project, Google Colab was chosen as the development environment due to its compatibility with deep learning libraries and support for GPU acceleration. The first step is to ensure that all necessary libraries are installed and imported.

The main libraries used include:

- **TensorFlow and Keras**: Used for building and training the CNN model.

- **Pandas and NumPy**: Enable data manipulation and analysis in tabular form.
- **NLTK (Natural Language Toolkit)**: Used for text preprocessing, including stopword removal and stemming.
- **Scikit-learn**: Provides tools for dataset splitting into training and test sets, as well as model evaluation.
- **Google Colab Files**: Allows uploading and loading files directly within the Colab environment.

Once these libraries are installed and imported, the environment is ready for data loading and preparation.

Loading the Arabic Dataset and Data Preparation: Arabic sentiment analysis requires a dataset containing textual reviews labeled according to their sentiment polarity (positive or negative). The dataset is stored in a `.tsv` file, which is manually uploaded into Google Colab.

A. Data Preprocessing

Once the dataset is loaded, several steps are necessary to clean and prepare it for use in the CNN model:

1) **Removing Special Characters**: This includes punctuation and symbols that do not provide meaningful information to the model.
2) **Tokenization**: The text is split into individual words for easier processing.
3) **Stopword Removal**: Arabic stopwords are filtered out to retain only meaningful words.
4) **Stemming**: Each word is reduced to its root form to group different variations of the same word under a single representation.

After preprocessing, the dataset labels are converted into numerical values (1 for positive reviews and 0 for negative reviews).

B. Dataset Description

For this project, we use an Arabic Sentiment Analysis dataset containing 60,000 user reviews written in Arabic. This dataset is specifically designed for training and evaluating sentiment classification models.

Dataset Details

- **Total number of instances**: The dataset consists of 60,000 Arabic text reviews.
- **Number of sentiment classes**: The dataset is labeled into two sentiment categories:
  - **Positive** (30,000 reviews): Includes satisfied and favorable user feedback.
  - **Negative** (30,000 reviews): Includes complaints and dissatisfaction.
- **Class distribution**: The dataset is balanced, with an equal number of positive and negative reviews. This ensures that no class dominates, reducing the need for data balancing techniques.
- **Types of features**:
  - **Raw text**: The Arabic review written by users.
  - **Labels**: A categorical sentiment label (Positive or Negative).
- **Diversity of textual data**: The dataset includes various types of textual data, such as book reviews, hotel and restaurant feedback, and general opinions about experiences. This variety ensures a broad representation of sentiment expressions across different domains.

VI. Using Kaggle

A. Create a Kaggle Account

1) Go to https://www.kaggle.com/ and sign up if you haven't already.
2) Verify your email and complete your profile.

B. Create a Kaggle Notebook

1) Navigate to `Code ¿ New Notebook`.
2) Rename your notebook, for example: `Arabic_Sentiment_Analysis`.

C. Using the Arabic Sentiment Analysis Module

You can use the publicly available module for Arabic sentiment analysis on Kaggle:

- Module: https://www.kaggle.com/code/oussamatouijer/arabic-sentiment-analysis

D. Dataset for Arabic Sentiment Analysis

For training and evaluation, you can use the following dataset:

- Dataset: https://www.kaggle.com/datasets/oussamatouijer/arabic-text-sentiment

E. Steps to Follow

By following these steps, you can:

- **Load and preprocess** your Arabic dataset.
- **Train a CNN model** on Kaggle.
- **Visualize and analyze** its performance.
- **Improve the model** by tuning hyperparameters.

Achieved Objectives::

- Loading and preprocessing data.
- Training a Deep Learning model.
- Analyzing results.
- Optimizing the model.

VII. Evaluation and Visualization of Results

This section presents the performance metrics used to evaluate the fine-tuned CNN model and the visualization of results to analyze its effectiveness.

Performance Metrics: To measure the model's performance in Arabic sentiment analysis, we use the following metrics:

Accuracy is a widely used metric to evaluate the performance of a classification model. It represents the percentage of correctly classified reviews out of the total predictions made by the model. In other words, it measures the proportion of correct predictions (both positive and negative) compared to the total number of predictions.

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}}$$

Precision measures the proportion of positive (or negative) reviews predicted by the model that are actually correct.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall measures the proportion of actual positive (or negative) reviews that the model correctly identified.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

F1-Score s is the harmonic mean of precision and recall, and is especially useful in cases of imbalanced datasets. The F1-score balances the trade-off between precision and recall, ensuring a model does well in both identifying relevant reviews (high recall) and making correct predictions (high precision).

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

VIII. Visualization of Training and Results

A. Evolution of Loss and Accuracy Over Epochs

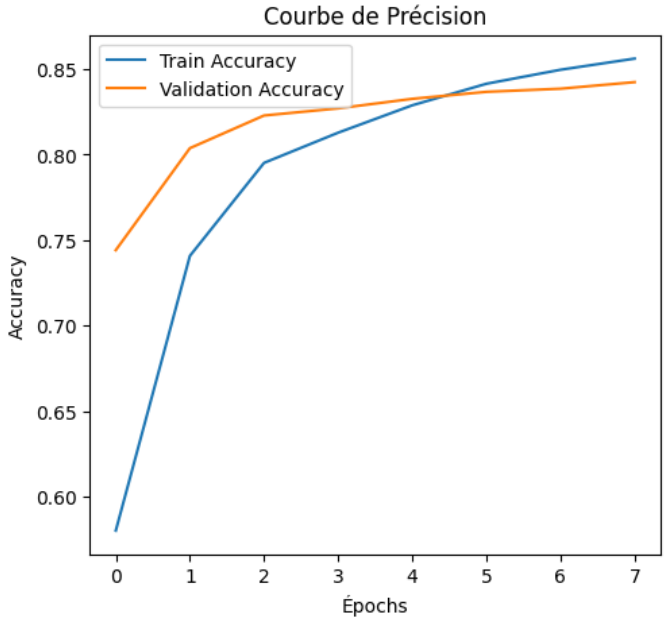

Figure 5. Accuracy Curve: Training and validation accuracy over epochs. The training accuracy steadily improves, reaching approximately 84%, while the validation accuracy stabilizes around the same value. The small gap between the curves indicates good generalization.

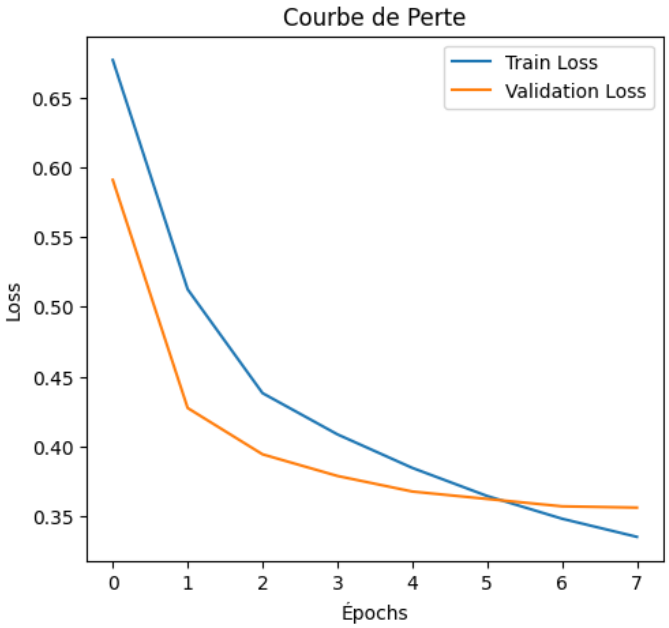

Figure 6. Loss Curve: Training and validation loss over epochs. The training loss decreases continuously, and the validation loss stabilizes after a few epochs. The absence of an increase in validation loss suggests no overfitting.)

Figure 7. Evolution of Loss and Accuracy Over Epochs

The graphs above illustrate the evolution of loss and accuracy over training epochs for both the training and validation datasets.

1) Accuracy Curve:

- The training accuracy steadily improves, reaching approximately 84% after several epochs.
- The validation accuracy also increases and stabilizes around 84%.
- The small gap between the training and validation accuracy curves indicates that the model generalizes well on unseen data.

2) Loss Curve:

- The training loss decreases continuously, indicating that the model is progressively learning the features of the dataset.
- The validation loss also follows a downward trend but stabilizes after a few epochs.
- The absence of an increase in validation loss suggests that the model does not suffer from overfitting.

B. Classification Report

	precision	recall	f1-score	accuracy
positive	0.84	0.83	0.84	0.84
negative	0.84	0.85	0.84	

Figure 8. Result

C. interpretation of result

The model performs well, but it could achieve better results with more specialized data. Currently, the data includes diverse topics like hotels, books, and opinions, which makes the learning process more challenging. A more focused dataset on a specific domain would allow the model to better capture relevant features. This would likely improve its overall performance and accuracy. Specialized data enhances the model's ability to generalize and predict more accurately.

IX. Conclusion

In this project, we successfully developed a Convolutional Neural Network (CNN) model to perform sentiment analysis on Arabic text data. The model demonstrated good performance, achieving approximately 84% accuracy on both the training and validation datasets. The evaluation metrics, including precision, recall, and F1-score, further confirmed the model's effectiveness in classifying sentiment correctly. However, the performance could be improved with more specialized and homogeneous data. Since the dataset used for training includes a variety of topics, such as hotels, books, and general opinions, the model had to learn from diverse contexts, which made it more challenging to capture domain-specific patterns. By focusing on a more targeted dataset, the model could better generalize and enhance its classification accuracy. Despite these limitations, the results show that the CNN model is effective for Arabic sentiment analysis and can be further fine-tuned with more specific data for improved outcomes.

References

[1] Kaggle Module: *Arabic Sentiment Analysis*. Retrieved from: https://www.kaggle.com/code/oussamatouijer/arabic-sentiment-analysis

[2] Kaggle Dataset: *Arabic Text Sentiment*. Retrieved from: https://www.kaggle.com/datasets/oussamatouijer/arabic-text-sentiment