Sentiment Analysis Using CNNs with Lexicon Integration and Attention Mechanisms

GitHub repo: https://github.com/hps-nguyen/lign167

Introduction

Sentiment analysis involves determining the sentiment polarity (e.g., positive or negative) of a given text. This project explores the use of convolutional neural networks (CNNs) for sentiment classification, focusing on their lightweight nature and suitability for this task. While modern NLP techniques often employ transformers with self-attention mechanisms, CNNs remain valuable due to their computational efficiency and ability to capture local patterns in text data. Drawing inspiration from "Lexicon Integrated CNN Models with Attention for Sentiment Analysis," this study evaluates the integration of external sentiment lexicons and attention mechanisms into CNN architectures to enhance performance.

The project compares a baseline CNN model with advanced variants that incorporate lexicon-based features and attention mechanisms. The objective is to assess the effectiveness of these enhancements in improving sentiment classification performance.

Methods

Convolution for Sentiment Analysis

CNNs in this project extract n-gram patterns efficiently, making them well-suited for sentiment analysis. By sliding convolutional filters over sequences, CNNs capture contextual patterns within windows defined by kernel sizes, such as bigrams and trigrams. Filters of varying sizes (e.g., 3, 4, 5) enable the model to simultaneously detect several n-gram patterns, enhancing its ability to analyze localized context. Max-pooling reduces the dimensionality of feature maps, preserving the most salient patterns while discarding less important ones.

While CNNs lack the sequential modeling of RNNs and the global context-awareness of transformers, they are computationally efficient and highly effective for tasks requiring localized context understanding, such as sentiment analysis. Their parallel processing capability allows faster training compared to RNNs.

Lexicon Embedding

Sentiment lexicons, such as SentiWordNet, provide external sentiment knowledge by mapping words to multi-dimensional sentiment scores. This project employs a 10-dimensional lexicon embedding for each word, encompassing positive, negative, and objective sentiment scores, part-of-speech encodings (noun, verb, adjective, adverb), and random trainable features. By concatenating lexicon features with word embeddings, the model incorporates both learned and handcrafted sentiment information.

For instance, the word "good" may have a learned embedding of [0.2, 0.5, 0.3] and a lexicon vector of [0.8, 0.0, 0.2, 1, 0, 0, 0, -0.3, 0.4, 0.5], resulting in a combined input of [0.2, 0.5, 0.3, 0.8, 0.0, 0.2, 1, 0, 0, 0, -0.3, 0.4, 0.5]. This integration, inspired by the original paper, enriches the input representation, improving the model’s sentiment classification capabilities.

Context-free Attention

The attention mechanism, derived from the original paper, dynamically assigns importance to tokens based on their relevance to the task. Unlike self-attention in transformers, which models interactions between tokens, context-free attention computes independent importance scores for each token using a fully connected (FC) layer. These scores are normalized via softmax, resulting in weights that prioritize significant tokens while diminishing less relevant ones. The weighted token representations are processed through convolutional layers, allowing the model to focus on key sentiment cues.

Models and Training

Datasets

The IMDb dataset, a widely used benchmark for sentiment analysis, is employed for training and evaluation. It contains 50,000 labeled movie reviews, divided into 40,000 for training and 10,000 for testing, with binary sentiment labels (positive or negative).

Text tokenization is performed using NLTK's TreebankWordTokenizer, and a vocabulary is constructed from these tokens. Sequences are padded or truncated to a fixed length of 512 tokens. Lexicon features are generated for each token using SentiWordNet, creating a 10-dimensional vector per word. Unknown words are assigned zero vectors. Each data point is a dictionary consisting of `tokens`—the vocabulary-encoded sequence, `lexicon_features`—the lexicon embedding of the sequence, and `label`—the sentiment label of the sequence. These processed inputs are used to train the models.

Baseline CNN

The baseline CNN consists of an embedding layer, convolutional layers with filters of sizes 3, 4, and 5, and max-pooling layers that extract salient features. The resulting features are concatenated and passed through a fully connected layer with a softmax activation for classification. This architecture efficiently captures local patterns but lacks external sentiment knowledge.

```
BaselineCNN(
  (embedding): Embedding(187679, 50)
  (convs): ModuleList(
    (0): Conv2d(1, 50, kernel_size=(3, 50), stride=(1, 1))
    (1): Conv2d(1, 50, kernel_size=(4, 50), stride=(1, 1))
    (2): Conv2d(1, 50, kernel_size=(5, 50), stride=(1, 1))
  )
  (fc): Linear(in_features=150, out_features=2, bias=True)
  (dropout): Dropout(p=0.5, inplace=False)
  (softmax): Softmax(dim=1)
)
```

Lexicon Integrated CNN

This model extends the baseline by incorporating external lexicon features. The lexicon embeddings are concatenated with word embeddings, creating a richer input representation. The convolutional layers are

adapted to process the increased dimensionality. This integration enables the model to capture sentiment-specific patterns more effectively.

```
LexiconIntegratedCNN(
  (embedding): Embedding(187679, 50)
  (convs): ModuleList(
    (0): Conv2d(1, 50, kernel_size=(3, 60), stride=(1, 1))
    (1): Conv2d(1, 50, kernel_size=(4, 60), stride=(1, 1))
    (2): Conv2d(1, 50, kernel_size=(5, 60), stride=(1, 1))
  )
  (fc): Linear(in_features=150, out_features=2, bias=True)
  (dropout): Dropout(p=0.5, inplace=False)
  (softmax): Softmax(dim=1)
)
```

Lexicon Integrated CNN with Attention

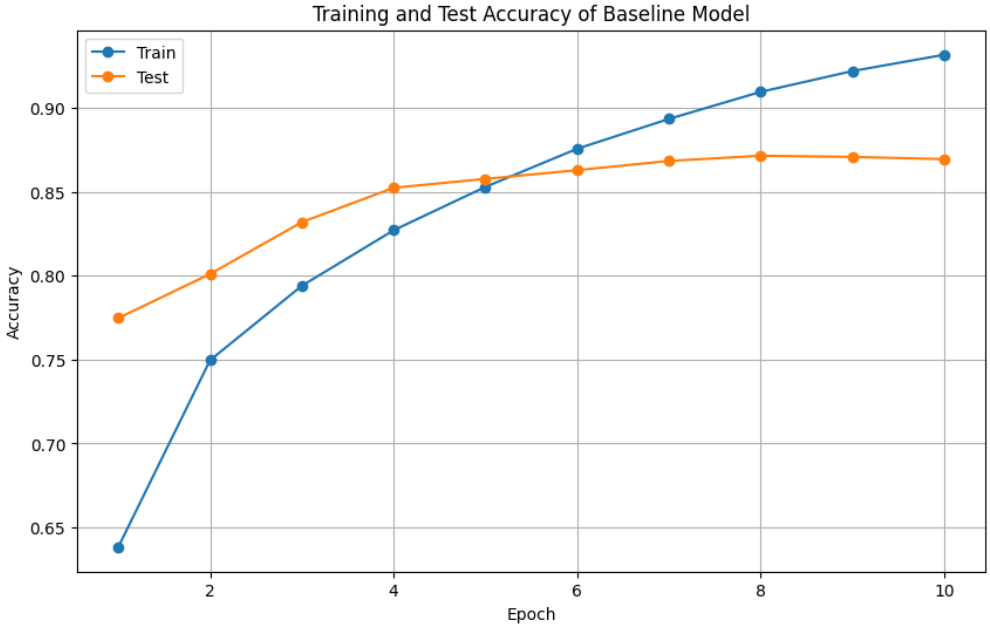
Building on the lexicon-integrated model, this variant employs an attention mechanism to dynamically prioritize tokens. An FC layer computes attention scores, which are normalized using softmax. These scores weight the token representations, emphasizing key sentiment cues before convolution. The architecture accommodates the combined dimensions of embeddings and lexicon features.

```
LexiconIntegratedAttentionCNN(
  (embedding): Embedding(187679, 50)
  (attention_fc): Linear(in_features=60, out_features=1, bias=True)
  (convs): ModuleList(
    (0): Conv2d(1, 50, kernel_size=(3, 60), stride=(1, 1))
    (1): Conv2d(1, 50, kernel_size=(4, 60), stride=(1, 1))
    (2): Conv2d(1, 50, kernel_size=(5, 60), stride=(1, 1))
  )
  (fc): Linear(in_features=150, out_features=2, bias=True)
  (dropout): Dropout(p=0.5, inplace=False)
)
```

Results and Discussion

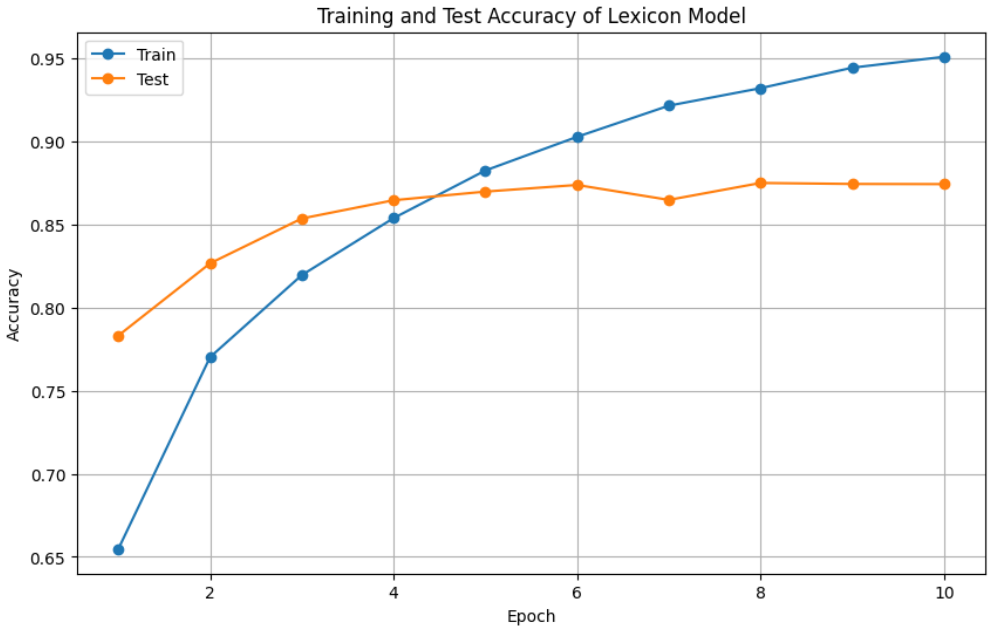
Baseline CNN

The baseline model achieved a test accuracy of 86.96% after 10 epochs. Its ability to capture n-gram patterns using convolutional filters contributed to its performance, but the lack of external sentiment knowledge limited its generalization.



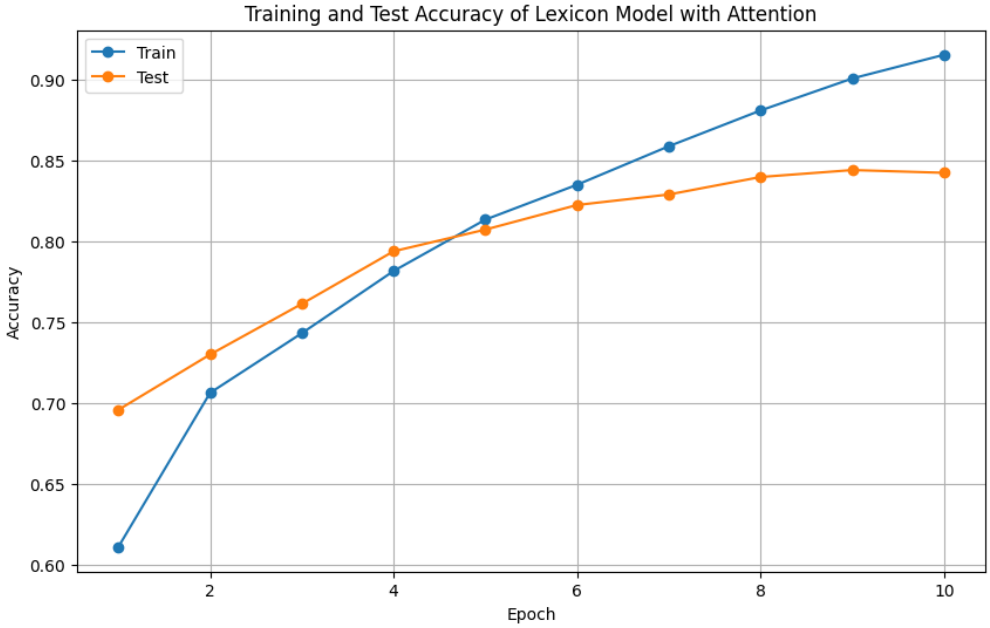
Lexicon Integrated CNN

With the addition of lexicon features, this model surpassed the baseline, achieving a test accuracy of 87.44%. The richer feature representation enabled better generalization and enhanced sentiment classification capabilities.



Lexicon Integrated CNN with Attention

While the attention-integrated model introduced dynamic token prioritization, its complexity resulted in slight overfitting, achieving a test accuracy of 83.54%. Given more training epochs and regularization techniques such as dropout and weight decay, this model could potentially surpass the lexicon-integrated variant.



Discussion

The Lexicon Integrated CNN demonstrated the best balance between efficiency and accuracy, significantly enhancing its ability to generalize on unseen data through the integration of additional sentiment information. The attention-based model introduced benefits in prioritizing key tokens but did not outperform the lexicon-integrated model due to its increased complexity. However, it showed potential for improvement in longer training regimes or with further regularization techniques such as dropout, weight decay, and learning rate adjustments. Hyperparameter tuning could also refine its performance. Unfortunately, this project could not fully optimize the attention-based model due to time and resource constraints, limiting its potential to meet expectations outlined in the original research.

Conclusion

This project demonstrated the effectiveness of enhancing CNNs with external lexicon features and attention mechanisms for sentiment analysis. The lexicon-integrated model provided the best balance between efficiency and accuracy. Although the attention-based model showed potential, resource constraints limited its optimization. Future work could focus on hyperparameter tuning and extended training to fully realize its benefits.