Homework3

LSTM + Attention

With Persian News Classification

Course:

Deep Learning

Supervisor:
Dr. Mansoori

Atefe Rajabi

Summer 2024

Index

Introduction

Natural Language Processing (NLP) is a pivotal domain within artificial intelligence that aims to enable computers to understand, interpret, and generate human language. This encompasses a variety of tasks including language translation, sentiment analysis, text summarization, and question answering. The advent of deep learning, particularly through architectures such as Long Short-Term Memory (LSTM) networks, has significantly propelled advancements in NLP. LSTMs are adept at capturing long-term dependencies in sequences, making them especially effective for processing and generating human language. They employ a gating mechanism to regulate the flow of information, thereby addressing challenges such as the vanishing gradient problem found in traditional Recurrent Neural Networks (RNNs).

Problem Definition

The objective of this project is to implement an LSTM model with an attention mechanism to classify a Persian news dataset. The dataset contains 96,430 samples across six different news categories: Political, Economic, Sports, Cultural/Artistic, Events, and Scientific news. Each sample includes a title, a subgroup (category), an abstract (short summary), and the body content of the news. The project involves several key tasks:

1. Data Preprocessing: Apply necessary preprocessing techniques to clean and prepare the text data for modeling. This includes handling missing values and deciding whether to remove stop words.

2. Text Representation: Investigate different methods for representing textual data, such as one-hot embeddings and the bag-of-words model, and choose the most appropriate one.

3. Model Implementation: Develop an LSTM model enhanced with an attention mechanism to improve classification performance.

4. Model Training and Evaluation: Train the model on the processed dataset and evaluate its performance. Save the trained model for future use.

Project Description

This project aims to classify Persian news articles into their respective categories using deep learning techniques. The model leverages the power of LSTM networks and attention mechanisms to effectively handle and analyze the textual data, addressing the unique challenges presented by the Persian language. The tasks include preprocessing the data, implementing the model architecture, and evaluating its performance on a large dataset of Persian news articles.

Implementation Detail

Preprocess

1. handle_missed:

This function ensures that any missing values in the 'abstract' and 'body' fields of the dataset are handled by filling them with empty strings. This prevents issues during text processing caused by missing data.

2. clean_text:

This function is designed to clean and sanitize the input text by performing several tasks:

- Replaces zero-width non-joiner characters with spaces.

- Removes URLs, email addresses, numbers, and English letters.

- Eliminates repeated characters.

- Removes special characters, leaving only alphanumeric characters and spaces.

- Trims extra spaces to ensure clean and consistent text.

3. normalization:

This function normalizes the text using the Hazm library, which helps to standardize the text by converting it to a consistent format. This is crucial for ensuring uniformity in the text data.

4. tokenization:

This function tokenizes the text using the Hazm library, which breaks the text into individual tokens (words).

5. find_stop_words:

This function identifies stop words based on their Inverse Document Frequency (IDF) scores using a TF-IDF vectorizer. Words with IDF scores outside a specified range are considered stop words. These identified stop words are then saved to a file for future use.

6. get_stop_words:

This function reads a file containing stop words (common words that are typically removed in text processing) and returns them as a set. These stop words are used to filter out non-informative words from the text data.

7. remove_stop_words:

This function removes stop words from a list of tokens, resulting in a list of filtered tokens that contain only the meaningful words. This step is essential for improving the quality and relevance of the text data for analysis.

8. lemmization:

The `lemmatizing` function processes a list of tokens by converting each token to its base or dictionary form using Hazm lemmatizer. This step ensures that the words are standardized, reducing variations and maintaining their contextual meaning.

9. Word Embedding

word embedding process using the Word2Vec model from Gensim:

The ‘Word2Vec’ model is created using the filtered tokens from the dataset. Key parameters include:

- ‘vector_size’: The size of the word vectors

document_vector mean Function:

This function generates a document vector by averaging the word vectors of the words in the document. The process is as follows:

- Filters out words that are not in the Word2Vec vocabulary.

- If the document contains any words, their vectors are averaged to create a single document vector.

- If no words are present, it returns a zero vector of the specified input shape (vector_size dimensions).

- This method is utilized.

document_vector_tfidf Function:

This function creates a TF-IDF weighted document vector. The steps are:

- Filters out words that are not in the Word2Vec vocabulary.

- Transforms the document words into TF-IDF weights using a TF-IDF vectorizer.

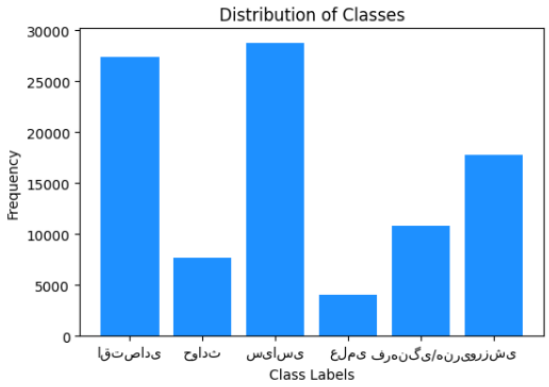
- Initializes a zero vector for the weighted average.

- Iterates over the document words, accumulating their weighted vectors and the total weight.

- If there are weights, it normalizes the weighted vector by the total weight.

- Returns the weighted document vector.

Data Imbalance:



From the distribution of the data, it is evident that there is a data imbalance. The SMOTE method was applied to the training data after embedding, but it did not yield good results.

Model Architecture

AttentionLayer Class

Purpose: The AttentionLayer is designed to focus on the most relevant parts of the input sequence, enhancing the performance of the model by weighting the importance of different parts of the input.

- 'self.W': A weight matrix initialized using the Glorot uniform initializer. This matrix will be used to compute the attention scores.

- 'self.b': A bias vector initialized to zeros. This bias is added to the attention score calculation.

- 'e': The raw attention scores calculated using a dot product of the input 'x' with the weight matrix 'W' and adding the bias 'b', followed by applying a tanh activation function.

- 'a': The normalized attention scores obtained by applying a softmax function to the raw attention scores along the sequence dimension.

- 'output': The input 'x' is multiplied by the attention scores 'a' to get the weighted output, and then summed across the sequence dimension.

Model Layer:

1. Input Layer:

Hidden_dim = data.shape[2]

Bidirectional LSTM Layer:

- 'Bidirectional(LSTM(hidden_dim, return_sequences=True, kernel_regularizer=l2(0.01), recurrent_regularizer=l2(0.01)))': This layer consists of an LSTM that processes the input sequences in both forward and backward directions. It returns sequences of hidden states for each time step. Regularization (L2) is applied to both the kernel and recurrent weights to prevent overfitting.

Dropout Layer:

- 'Dropout(0.2)': This layer randomly drops 20% of the neurons during training, which helps in preventing overfitting.

Attention Layer:

- 'AttentionLayer()': This custom layer computes the attention scores for the LSTM output, weighs the hidden states by these scores, and summarizes them to produce a context vector.

Dense Layer:

- 'Dense(hidden_dim//2, activation='relu')': This fully connected layer reduces the dimensionality of the attention output by a factor of 2 and applies a ReLU activation function.

Batch Normalization Layer:

- 'BatchNormalization()': This layer normalizes the outputs of the previous layer to improve the training stability and speed.

Dense Layer:

- ‘Dense(hidden_dim//4, activation='relu')’: This fully connected layer reduces the dimensionality of the attention output by a factor of 4 and applies a ReLU activation function.

Output Dense Layer:

- ‘Dense(len(label_encoder.classes_), activation='softmax')’: The final output layer produces a probability distribution over the target classes using the softmax activation function.

‘model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])’: The model is compiled with the Adam optimizer and sparse categorical cross-entropy loss function, which is suitable for multi-class classification problems with integer labels. The accuracy metric is used to evaluate the model's performance.

‘EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)’: This callback monitors the validation loss during training and stops training if the loss does not improve for 3 consecutive epochs. It also restores the model weights to the best observed values.

Model Architecture:

Layer (type)	Output Shape	Param #
bidirectional_24 (Bidirectional)	(None, 1, 1024)	4,198,400
dropout_24 (Dropout)	(None, 1, 1024)	0
attention_layer_24 (AttentionLayer)	(None, 1024)	1,025
dense_51 (Dense)	(None, 256)	262,400
batch_normalization_24 (BatchNormalization)	(None, 256)	1,024
dense_52 (Dense)	(None, 128)	32,896
dense_53 (Dense)	(None, 6)	774

Total params: 4,496,519 (17.15 MB)

Trainable params: 4,496,007 (17.15 MB)

Non-trainable params: 512 (2.00 KB)

Results

Train and validation Loss

precision	recall	F1-score	support	label
0.92	0.90	0.91	5614	اقتصادی
0.91	0.92	0.92	1503	حوادث
0.89	0.91	0.90	5653	سیاسی
0.80	0.78	0.79	821	عملی
0.89	0.91	0.90	2197	فرهنگی / هنری
0.98	0.97	0.98	3498	ورزشی

accuracy			0.91	19286
macro avg	0.90	0.90	0.90	19286
weighted avg	0.91	0.91	0.91	19286