Kriti 2024 - ML AI Problem Statement

Understanding Text Classification

What is Text Classification?

- The process of categorizing text into organized groups.
- Utilizes Natural Language Processing (NLP), machine learning, and deep learning.

Key Applications:

1. **Sentiment Analysis**
2. **Topic Labeling**
3. **Spam Detection**
4. **Language Detection**
5. **Intent Recognition**
6. **Automated Customer Support**

Long Short-Term Memory Networks (LSTMs)

- A special kind of Recurrent Neural Network (RNN) capable of learning long-term dependencies.
- Designed to avoid the long-term dependency problem typical in standard RNNs.

Key Features :

- Memory Cells
- Gates
- Long-Term Dependencies

CHAPTER TITLE

Why LSTMs for Text?

Sequential Data Processing	Naturally suited for text, where the sequence of words and context significantly impacts meaning.
Versatility	Applied in a range of NLP tasks like sentiment analysis, language translation, and more.
Improved Performance	Outperforms traditional RNNs in learning and remembering over long sequences, making it ideal for complex text processing tasks.

CHAPTER TITLE

Preparing the Dataset

```
# Load data
train_df = pd.read_csv("/kaggle/input/disanggirls/train.csv")
test_df = pd.read_csv("/kaggle/input/disanggirls/test.csv")
sub_format=pd.read_csv("/kaggle/input/disanggirls/sample_submission.csv")
# Combine title and abstract
train_df['text'] = train_df['Title'] + ' ' + train_df['Abstract']
test_df['text'] = test_df['Title'] + ' ' + test_df['Abstract']
```

Overview of Dataset Structure

- Components: Text data is segmented into titles, abstracts, and categories.
- Objective: Combine textual components for a comprehensive analysis and categorize them accordingly.

Loading the Data

- Tools: Utilize pandas, a Python library, for efficient data handling.
- Files: Data is divided into training (train.csv), testing (test.csv), and submission format (sample_submission.csv).

Preprocessing Steps

- Combination: Merge 'Title' and 'Abstract' to enrich the dataset, forming a singular text column per entry.
- Purpose: This unified text column enhances the model's ability to learn from both the concise titles and detailed abstracts.

CHAPTER TITLE

Text Preprocessing: Tokenization and Padding

```
# Padding sequences
max_length = max([len(seq) for seq in train_sequences])
train_padded = pad_sequences(train_sequences, maxlen=max_length)
test_padded = pad_sequences(test_sequences, maxlen=max_length)
```

```
# Tokenization
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(train_df['text'])

train_sequences = tokenizer.texts_to_sequences(train_df['text'])
test_sequences = tokenizer.texts_to_sequences(test_df['text'])
```

Text Tokenization

- Definition: Process of converting text into a sequence of integers, with each integer representing a unique word in the dataset.
- Tool Used: Keras Tokenizer.
- Purpose: Streamlines text data to fit the model by representing words as numerical values.
- Efficiency: Limits the vocabulary to the top 5000 words to manage memory and computational resources effectively.

Sequence Padding

- Objective: Standardize the length of all text sequences for consistent model input.
- Method: Pad shorter sequences with zeros to match the length of the longest sequence in the dataset.
- Rationale: Ensures that the model receives input of uniform size, crucial for training stability and performance.

Handling Multi-label Classification

```
# Convert categories to one-hot encoding
# encoder = LabelBinarizer()
# train_labels = encoder.fit_transform(train_df['Categories'])
mlb = MultiLabelBinarizer()
train_df['Categories'] = train_df['Categories'].apply(eval)  # Convert string representations to lists
train_labels = mlb.fit_transform(train_df['Categories'])
```

Multi-label Classification Defined

- Concept: Unlike single-label classification where each text is assigned to one category, multi-label classification allows each text to belong to multiple categories simultaneously.
- Importance: Reflects real-world complexities where content often spans multiple themes or subjects.

The Role of MultiLabelBinarizer

- Function: Transforms lists of category labels into a binary matrix representation.
- Purpose: Each column represents a unique label, and each row represents a text instance. A "1" indicates the presence of the label for the given text, while "0" signifies absence.
- Advantage: Simplifies processing and modeling of multi-label data by converting it into a format that machine learning models can understand and work with.

Application in Deep Learning

- Model Adaptation: Models can be adapted to predict multiple outputs simultaneously, making them capable of handling the intricacies of multi-label classification tasks.
- Evaluation: Requires metrics that can assess performance across multiple labels, such as hamming loss, precision, recall, and F1 score for a comprehensive evaluation.

LSTM Model Construction

```
model = Sequential()
model.add(Embedding(input_dim=5000, output_dim=50, input_length=max_length))
model.add(SpatialDropout1D(0.2))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
#model.add(Dense(train_labels.shape[1], activation='softmax'))
model.add(Dense(len(mlb.classes_), activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
#model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Embedding Layer

- Purpose: Converts integer sequences (words) into dense vectors of a fixed size, facilitating the model's understanding of text as continuous, meaningful representations.
- Benefits: Reduces dimensionality and captures semantic relationships between words.

SpatialDropout1D

- Function: Drops entire 1D feature maps instead of individual elements, addressing overfitting by simplifying the model and making it more robust to noise in the input data.
- Advantage: Particularly effective in NLP tasks to prevent over-reliance on specific word patterns.

LSTM Layer

- Role: Processes the sequence input, capturing long-term dependencies between word occurrences in the text. It's key for understanding context and the sequence in which words appear.
- Capabilities: LSTMs can remember information for an extended number of time steps, making them ideal for processing and making predictions based on long text sequences.

Dense Output Layer

- Function: Outputs a probability distribution over the label space, indicating the likelihood of each category being applicable.
- Activation Function: Uses sigmoid activation for multi-label classification, allowing for independent classification probabilities for each label.

Efficient Model Training

```
batch_size = 64
epochs = 5

model.fit(train_padded, train_labels, epochs=epochs, batch_size=batch_size, validation_split=0.1)
```

Key Training Parameters

- Purpose: Converts integer sequences (words) into dense vectors of a fixed size, facilitating the model's understanding of text as continuous, meaningful representations.
- Benefits: Reduces dimensionality and captures semantic relationships between words.

Validation for Performance Monitoring

- Validation Split (e.g., 10%): Reserves part of the training data to evaluate model generalization, crucial for tuning and preventing overfitting.

Overfitting Mitigation Strategies

- Early Stopping: Halts training when validation performance declines, preserving model generalizability.
- Dropout & Regularization: Techniques to simplify the model, ensuring it learns robust patterns.

Efficient Training Insights

- Careful management of epochs and batch size, alongside validation monitoring, optimizes learning and model reliability.
- Adopting early stopping and regularization guards against overfitting, enhancing model's real-world applicability.

Making Predictions & Preparing Submissions

```
predictions = model.predict(test_padded)
# Convert predictions to binary labels based on a threshold (e.g., 0.5)
binary_predictions = (predictions > 0.5).astype(int)

# Use inverse_transform to convert the binary labels back to the original class labels
predicted_classes = mlb.inverse_transform(binary_predictions)

# Convert predictions to binary labels
#predicted_labels = (predictions > 0.5).astype(int)
# Create submission DataFrame
# submission_df = pd.DataFrame(predicted_labels, columns=mlb.classes_)
# submission_df.insert(0, 'Id', test_df['Id'])
# submission_df.to_csv('submission_corrected.csv', index=False)
```

```
ids=test_df["Id"]
#m={}
# for cl in sub_format.columns:
#     m[cl]=0
final_output=pd.DataFrame(columns=sub_format.columns)
```

```
# Convert the accumulated rows into a DataFrame
final_output = pd.DataFrame(to_add, columns=sub_format.columns)

# Save the DataFrame to a CSV file
final_output.to_csv('submission_corrected11.csv', index=False)
```

Prediction Process

- Testing Data: Apply the trained model to unseen test dataset to generate predictions.
- Binary Conversion: Model outputs are probabilities. Apply a threshold (e.g., 0.5) to convert these into binary labels (1 for presence, 0 for absence of a category).

Mapping Predictions to Categories

- Inverse Transform: Use MultiLabelBinarizer.inverse_transform to convert binary labels back to original categorical labels, ensuring predictions are interpretable.

Preparing Submission File

- Format Matching: Align predicted categories with the competition's submission format, typically requiring an ID and predicted categories per entry.

Conclusion & Future Directions

Project Recap

- Executed a comprehensive workflow: Data preprocessing, LSTM model design, training, and multi-label prediction.
- Highlighted LSTM's prowess in complex text classification, effectively managing long sequences and multiple labels.

Future Enhancement Strategies

- Explore Architectures: Test different neural networks (e.g., GRUs, Transformers) for potential gains.
- Hyperparameter Optimization: Employ advanced tuning to refine model performance.
- Data Augmentation: Enrich training with varied datasets or additional textual features to improve context understanding.

Closing Note

Our exploration underscores LSTM's significant role in NLP, setting the stage for further innovation and enhanced model sophistication in text analysis.

Presentation By :
Disang (Girls)