Overview

The notebook processes Quora question text data to build a binary classification model (target `0` or `1`) using:

- **Text preprocessing** with NLTK (lemmatization, stopword/punctuation removal)
- **GloVe word embeddings** (300-dimensional vectors from `glove.42B.300d.txt`)
- **LSTM architecture** for sequence modeling

Key Components

1. Data Preparation

```
df = pd.read_csv('/content/drive/MyDrive/Datasets/Quora Text Classification Data.csv')
df['Clean Text'] = df['question_text'].progress_apply(cleaning)
```

- Input: CSV file with `question_text` (raw text) and `target` (binary label)
- Cleaning pipeline:
   a. Lowercase conversion
   b. Tokenization using `word_tokenize`
   c. Stopword/punctuation removal
   d. Lemmatization with `WordNetLemmatizer`

2. Embedding Layer Setup

```
!unzip '/content/drive/MyDrive/Word Embeddings/glove.42B.300d.zip'
embedding_matrix = np.zeros((vocab_size,300))
for word, i in tokenizer.word_index.items():
    embedding_matrix[i] = embedding_values.get(word, np.zeros(300))
```

- Loads 300D GloVe vectors into dictionary
- Creates embedding matrix mapping tokenized words to vectors

- Unknown words initialize as zero vectors

3. Model Architecture

```
model = Sequential()
model.add(Embedding(vocab_size,300,input_length=300,weights=[embedding_matrix],trainable=False))
model.add(LSTM(50))
model.add(Dense(128,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
```

- **Embedding Layer**: Uses frozen GloVe vectors
- **LSTM Layer**: 50 units for sequence processing
- **Dense Layers**: 128-unit ReLU + sigmoid output

Compilation:

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

Training Configuration

```
history = model.fit(pad_seq,y,validation_split=0.2,epochs=5)
```

- **Input**: Padded sequences (max length=300 tokens)
- **Validation**: 20% split from training data
- **Epochs**: 5 training iterations

Critical Design Choices

1. **Sequence Length**: Truncates/pads texts to 300 tokens
2. **Embedding Freezing**: Preserves pretrained GloVe semantics
3. **LSTM Configuration**: Balances complexity and performance

---

Optimization Opportunities

1. **Class Imbalance**: Add `class_weight` parameter if targets are skewed
2. **Bidirectional Processing**: Replace `LSTM` with `Bidirectional(LSTM)`
3. **Regularization**: Incorporate Dropout layers to reduce overfitting
4. **Hyperparameter Tuning**: Test different sequence lengths/unit counts

---

Execution Notes

1. Requires Google Drive access for dataset/embedding files
2. Uses Colab's free GPU acceleration (enable via *Runtime > Change runtime type*)
3. Preprocessing leverages NLTK's linguistic resources (`stopwords`, `WordNetLemmatizer`)

This implementation provides a robust baseline for binary text classification tasks using transfer learning from pretrained embeddings and sequential modeling via LSTMs.

⁂