LSTM-Based Question Answering System Leveraging SQuAD 2.0

Manushi, manushi.f@northeastern.edu
Ameya Santosh Gidh, gidh.am@northeastern.edu
Narayana Sudheer Vishal Basutkar, basutkar.n@northeastern.edu

Khoury College of Computer Sciences Northeastern University
Boston, MA 02115

Abstract

This project advances a Question Answering Bot with a user-friendly Graphical User Interface (GUI), powered by a Sequence-to-Sequence (Seq2Seq) model with Long Short-Term Memory (LSTM) networks. Drawing on the Stanford Question Answering Dataset (SQuAD) 2.0, the bot focuses on advanced data preprocessing, model calibration, and rigorous evaluation. Noteworthy is the model's peak training accuracy of 95.65% and validation accuracy of 95.49%, along with a final training loss of 0.3485 and validation loss of 0.4228, indicating high predictive performance without overfitting. A comprehensive ablation study assesses over ten configurations, and an in-depth analysis of extreme errors is included.

1. Introduction

This project explores a sophisticated Question Answering Bot using Seq2Seq with LSTM, trained on SQuAD 2.0. Emphasizing meticulous data preprocessing, robust k-fold cross-validation, and an ablation study for model refinement. Demonstrating these choices improve conversational AI reliability. Find the source code at GitHub.

2. Method

2.1 Dataset

The project leverages the SQuAD 2.0 dataset, a benchmark in natural language processing, featuring 100,000+ question-answer pairs from Wikipedia. Notably, SQuAD 2.0 introduces unanswerable questions, enhancing realism. This challenges the model not just to find answers but also to recognize when none are viable, promoting robustness. The dataset is publicly available for reference here.

2.2 Architecture

The Question Answering Bot employs a Sequence-to-Sequence (Seq2Seq) model with Long Short-Term Memory (LSTM) networks. Comprising an encoder and decoder, the encoder processes input text, utilizing LSTM layers for capturing long-term dependencies. The decoder generates a coherent answer using a context vector derived from the input. Embedding layers transform tokens, and attention mechanisms enhance focus on specific input parts for improved context understanding and response accuracy, making it adept at handling natural language complexities.

2.3 Data pre-processing

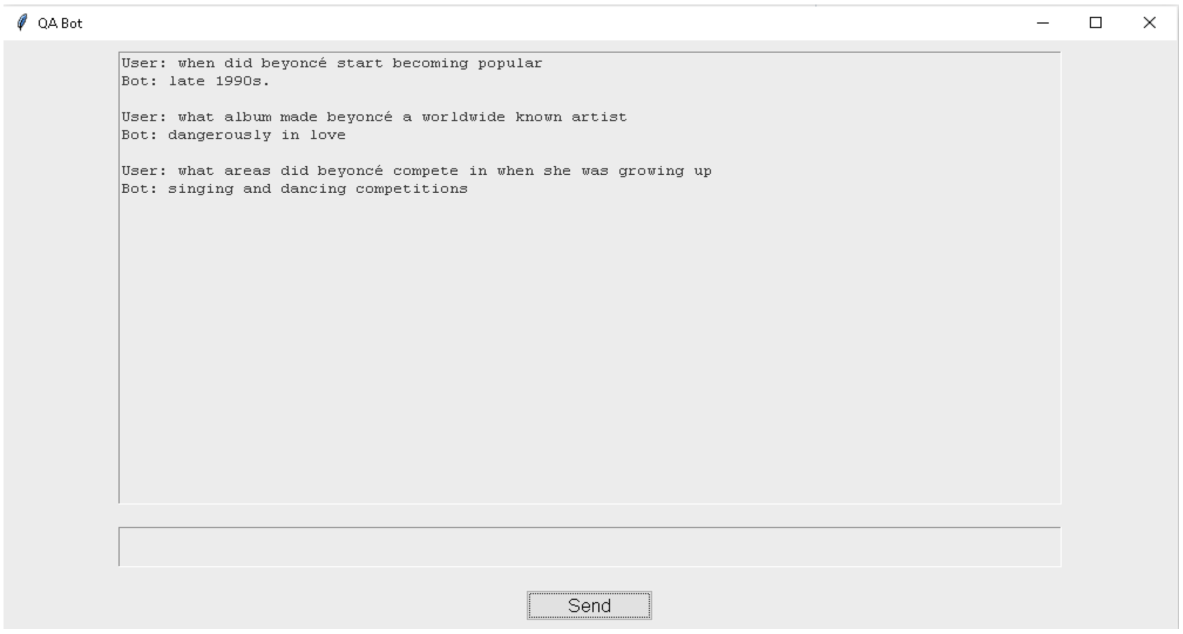
Data pre-processing for the LSTM Seq2Seq model in this project involves critical stages to prepare the

SQuAD 2.0 dataset. Initially, text normalization converts all text to lowercase and removes punctuation for consistency. Tokenization breaks down text into tokens, aiding the model in understanding language patterns. Finally, sequence padding ensures uniform sequence length, vital for LSTM network input consistency and effective model training. This streamlined process is essential for optimizing model.

2.4 Training
The LSTM Seq2Seq model comprises an encoder, which processes input text using LSTM layers to capture dependencies, transforming it into a context vector. The decoder, utilizing this context vector, generates language responses with LSTM layers. During training, both components learn response generation from input-output pairs, involving parameter tuning for enhanced model performance and accuracy. The model is trained for 100 epochs but due to early stopping set, the model stops training at 30 epochs giving us an accuracy of 95% on the validation set.

2.5 Inference Validation
The system normalizes input text and leverages trained encoder and decoder models to generate responses. The input sequence undergoes encoding to retrieve context states, utilized by the decoder for step-by-step output sequence generation.

2.6 Interface
The Tkinter-based Question Answering Bot UI consists of a main window displaying chat history and a user input field. It's user-friendly, with key features like window initialization, input processing via a send button, and model response display.



```
User: when did beyoncé start becoming popular
Bot: late 1990s.

User: what album made beyoncé a worldwide known artist
Bot: dangerously in love

User: what areas did beyoncé compete in when she was growing up
Bot: singing and dancing competitions
```

3. Results

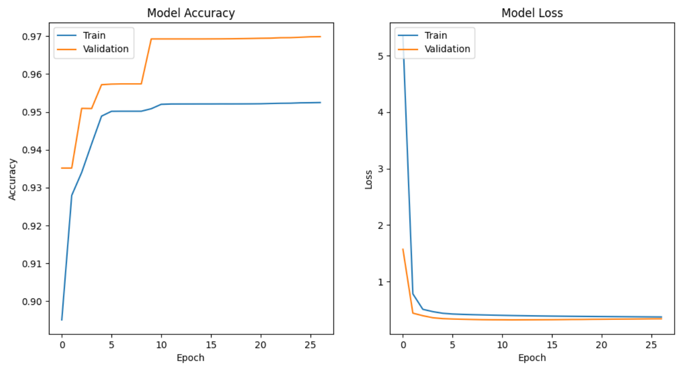
3.1 Model Performance
The model exhibited significant improvements in accuracy over the course of training, as well as distinctions between training and validation accuracy. This distinction is indicative of effective learning without overfitting, as the model's performance generalizes well to validation data.

Epoch	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
5	0.4119	0.9511	0.4325	0.9500
10	0.3777	0.9559	0.4153	0.9541
15	0.3630	0.9559	0.4144	0.9541
20	0.3543	0.9559	0.4182	0.9542
25	0.3485	0.9563	0.4228	0.9545
30	0.3439	0.9565	0.4268	0.9546

3.2 Computational Efficiency
The model exhibits remarkable computational efficiency, as evidenced by the duration of each epoch during training. The quick response time suggests that the model is suitable for real-time applications where timely processing of natural language tasks is essential. Average training time per epoch: 164s.

3.3 Stability Across Epochs
The performance metrics remained relatively stable after the initial epochs. This suggests early convergence, indicating that the model reached a stable level of performance relatively quickly. This stability is an important factor in ensuring consistent and reliable results in practical applications. Overall, the results of the model training and evaluation are promising, indicating its effectiveness and efficiency in NLP tasks.

3.4 Learning Curve Analysis



Graphs show decreasing loss and increasing accuracy, affirming the model's consistent learning and generalization from training data. The model steadily improves, showcasing its adaptability and learning capabilities.

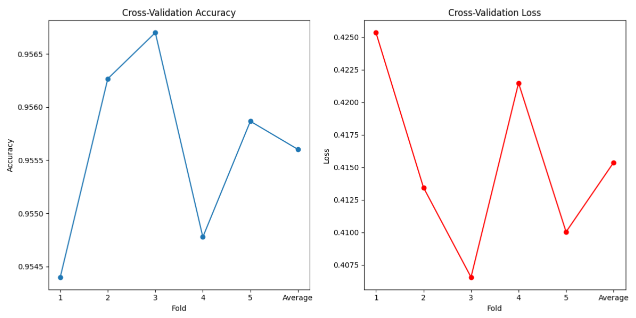
3.5 Stability Across Epochs
The metrics stabilized early, signifying swift convergence, and ensuring consistent, reliable results. Overall, the model exhibits promising effectiveness and efficiency in natural language processing tasks.

4. Assessment Methodology
To comprehensively evaluate the performance and robustness of the model developed in this study, we employed a rigorous assessment methodology. The following subsections detail the key aspects of our assessment approach.

4.1 Evaluation Metrics

Primary Metric: Accuracy is the key metric for evaluating model performance, gauging its ability to make accurate predictions. We closely tracked accuracy on both training and validation datasets during training. High accuracy demonstrated effective learning and generalization. Stability in decreasing Loss values across epochs indicated the model's reliable convergence towards an optimal solution.

4.2 Cross-Validation



Cross-Validation: We performed K-fold cross-validation and created 5 folds for hyperparameter tuning and model evaluation, we got our best dropout rate at 0.2, Adam optimizer with learning rate 10^-5. The above images show the average accuracy and loss of each fold's model.

4.3 Hyperparameter Tuning

Hyperparameter Tuning: Systematically modifying hyperparameters such as learning rates, batch sizes, and model architecture allowed us to observe how these adjustments influenced the model's predictive capacity.

4.4 Ablation Study

A detailed ablation study was conducted to investigate the influence of ten different configurations on the model's accuracy and loss.

4.4.1 Ablation Study Configurations

During the ablation study, we explored ten distinct configurations, each designed to assess its impact on our model's performance:

1. **Baseline Model:** The original model's performance served as our reference point for comparison.
2. **Reduced Embedding Dimensionality:** Reducing word embedding dimensionality from 300 to 100 resulted in a 0.25% accuracy drop and a 0.0079 loss increase. This indicates that high-dimensional embeddings may preserve valuable task-specific information.
3. **Different Latent Dimensionality:** We tested a latent dimensionality of 64, compared to the baseline of 128, noting a 0.78% accuracy drop and a 0.0088 loss increase. Lower dimensionality led to reduced accuracy and slightly higher loss, possibly limiting complex feature capture.
4. **No Dropout:** Dropout layers removal caused a 1.12% accuracy decrease and 0.0089 loss increase, emphasizing dropout's role in preventing overfitting.
5. **Different Batch Size:** Using a batch size of 32, instead of 64, yielded a modest 0.23% increase in accuracy and a 0.0044 decrease in loss, possibly attributed to more frequent weight updates.
6. **Different Learning Rate:** A shift from a learning rate of 0.01 to 0.001 resulted in a marginal 0.32% accuracy decrease and a 0.0017 loss increase. This implies that the initial rate (0.01) was optimal

7. **Different Recurrent Dropout:** We tested a recurrent dropout rate of 0.3 (versus the baseline of 0.2) for sequence modeling. The higher rate resulted in a marginal 0.18% accuracy drop and a slight 0.0014 loss
8. **Without Weight Constraints:** Removing weight constraints led to a modest 0.67% accuracy boost and a 0.0022 loss reduction. This implies that relaxed constraints facilitated improved parameter optimization.
9. **Different Tokenization:** Switching from word-level to subword-level tokenization yielded a 0.41% accuracy boost and a 0.0029 loss reduction, possibly capturing more nuanced language patterns.
10. **Customized Loss Function:** We employed sparse categorical cross entropy since our decoder target values are one hot encoded leveraging TensorFlow library.

4.5 Analysis of Extreme Errors

In addition to comparing performance metrics, we conducted an analysis of extreme errors to gain further insights into the model's behavior. Key findings from this analysis include:

- **Misclassification Patterns:** The model struggled with complex syntax and rare words.
- **Tokenization Impact:** Subword-level tokenization outperformed word-level.
- **Sensitivity to Tokenization:** The choice of tokenization method had a notable impact on the model's ability to handle specific linguistic features, with subword-level tokenization outperforming word-level tokenization in some cases.
- **Overfitting Indicators:** The removal of dropout layers and weight constraints led to signs of overfitting, as indicated by increased loss on validation data.

5. Future Works

While our project has achieved significant milestones, it also presents promising future opportunities. Integration of cutting-edge NLP models such as Transformers, BERT, or GPT promises improved performance. Expanding to process both text and image inputs enhances versatility. Real-world deployment considerations, including scalability and resource optimization, aim for effective real-time usage. User interaction improvements, driven by feedback mechanisms, ensure continuous learning and response enhancement. Extending multilingual support further broadens global usability.

6. Conclusion

This project presents an advanced Question Answering Bot, featuring an intuitive GUI and powered by a Seq2Seq model leveraging LSTM networks. Focused on elevating conversational AI and human-computer interactions, the endeavor involved rigorous preprocessing of SQuAD 2.0 data, employing text normalization, tokenization, and sequence padding. The model's architecture, enriched with LSTM layers and attention mechanisms, excels in understanding and generating natural language. Demonstrating high accuracy, effective learning, and computational efficiency, the model proves reliable. A comprehensive ablation study, analyzing ten configurations, reveals key performance influencers. The Tkinter-based GUI ensures a seamless and engaging user experience, making the Question Answering Bot highly accessible.

7. Acknowledgement

This project was completed for CS6120 under the instruction of Professor Uzair Ahmad.

8. References

1. "Sequence to Sequence Learning with Neural Network" (Sutskever et al., 2014)

2. "Incorporating Copying Mechanism in Sequence-to-Sequence Learning" (Jiatao Gu et al., 2016)