DSGA 1011: Assignment 4

Full Name: Shruti Tulshidas Pangare
Net ID: stp8232

Q0. 1.

Please provide a link to your github repository, which contains the code for both Part I and Part II. https://github.com/shrutipangare/Fine-tuning$_{L}anguage_{M}odels$

Q2. 1.

Describe your transformation of dataset.

For this assignment, I designed a synonym replacement transformation strategy that introduces linguistic variations while preserving the original semantic meaning and sentiment of the text.

Text Preprocessing

- Utilizes NLTK's advanced tokenization capabilities
- Converts entire text to lowercase for uniform processing
- Breaking text into individual word tokens
- Apply part-of-speech (POS) tagging to understand grammatical context

Word-Level Transformation Strategy

- Implemented 50% probabilistic word replacement mechanism
- Identify words eligible for potential synonym substitution
- Preserves grammatical structure during transformation

Synonym Selection Process

- Leverage WordNet semantic network
- Convert POS tags to WordNet-compatible format
- Find semantically appropriate synonyms
- Ensure single-word replacement preference, avoiding replacing the original word with itself

Transformation Examples

Example 1

- **Original:** ″The movie was incredibly exciting and fast-paced″
- **Transformed:** ″The film was immensely thrilling and rapid″

Converted ″movie″ to ″film″, ″incredibly″ to 'immensely', maintainign core semantic meaning

Example 2

- **Original:** "I was truly and wonderfully surprised..."
- **Transformed:** "i be truly and wonderfully surprised..."

Lowercase conversion, grammatical variation with "be", Preserved emotional sentiment

The transformation is "reasonable" because it:

- Simulates natural language variation
- Reflects how humans might rephrase similar ideas
- Introduces subtle linguistic diversity
- Represents plausible writing style variations

The transformation approach leverages sophisticated natural language processing techniques to introduce controlled linguistic variations while maintaining the fundamental semantic integrity of the original text. By systematically replacing words with their contextually appropriate synonyms, the method creates out-of-distribution data that reflects the nuanced ways human language can be expressed.

The proposed transformation mimics natural language variations that could emerge from different writing styles, individual vocabulary preferences, or contextual communication adaptations. It simulates how the same sentiment or idea can be articulated through diverse linguistic representations, thereby challenging machine learning models to demonstrate robust generalization capabilities.

Q3. 1

Report & Analysis

- Report the accuracy values for both the original and transformed test data evaluations.
  Original Test Data Accuracy: **0.930906 = 93%**
  Transformed Test Data Accuracy: **0.9056 = 90.56%**
  The augmentation technique effectively increased the model's robustness By exposing the model to linguistically transformed examples during training, it learned to generalize better The transformed examples likely helped the model develop more resilient feature representations
- Analyze and discuss the following: (1) Did the model's performance on the transformed test data improve after applying data augmentation? (2) How did data augmentation affect the model's performance on the original test data? Did it enhance or diminish its accuracy?
  The augmentation technique effectively increased the model's robustness, by exposing the model to linguistically transformed examples during training, it learned to generalize better, the transformed examples likely helped the model develop more resilient feature representations
- Offer an intuitive explanation for the observed results, considering the impact of data augmentation on model training.
  its like imagining training a language model like teaching a student to understand a language's essence beyond just memorizing specific phrases. The data augmentation is analogous to exposing the student to different ways of expressing the same idea: Original training: Learning specific sentences Augmented training: Learning the underlying semantic structure Result: More robust understanding that can handle varied linguistic expressions
- Explain one limitation of the data augmentation approach used here to improve performance on out-of-distribution (OOD) test sets.

The current data augmentation strategy is constrained by its narrow and simplistic transformation mechanism. By relying solely on synonym replacement with only 25% of words potentially transformed, the approach fails to capture the comprehensive complexity of linguistic variations. This limited approach introduces several critical limitations: it provides a superficial view of language diversity, potentially creates artificial training signals that may not reflect real-world linguistic nuances, and restricts the model's ability to develop truly robust generalization capabilities. The transformation is mechanistic and lacks the organic complexity of natural language variations, essentially creating a somewhat predictable and constrained set of augmented examples that may not sufficiently challenge the model to develop deep, contextually sensitive understanding. Moreover, the fixed probability of word replacement and reliance on wordnet-based synonyms means that the augmentation lacks the rich, contextual adaptability required to simulate the genuine linguistic diversity encountered in out-of-distribution scenarios.

Part II. Q4

Statistics Name	Train	Dev
Number of examples	4225	466
Mean sentence length	18.09	18.07
Mean SQL query length	217.37	211.05
Vocabulary size (natural language)	791	465
Vocabulary size (SQL)	555	395

Table 1: Data statistics before any pre-processing

Statistics Name	Train	Dev
T5 fine-tuned model		
Model Name	t5-small	t5-small
Mean sentence length(tokens) with prefix	22.09	22.07
Mean SQL query length(tokens) with prefix	119.67	118.77
Vocabulary size (natural language)	794	468
Vocabulary size (SQL)	555	395

Table 2: Data statistics after pre-processing

Q5

Design choice	Description
Data processing	The preprocessing stage involved meticulous text cleaning and standardization. By stripping unnecessary whitespace, the team ensured consistent input formatting. The crucial addition of the "translate to SQL:" prefix to natural language queries provides explicit contextual guidance to the model, helping it understand the specific translation task
Tokenization	The maximum source length of 512 tokens allows comprehensive input representation, while limiting the target length to 128 tokens prevents excessive output generation. Special tokens were strategically added to enhance the model's understanding of input-output boundaries and task-specific semantics
Architecture	Fine-tuned the T5-small model, providing a balanced computational strategy. The freezing of the first 2 encoder layers introduces a nuanced transfer learning technique. By potentially constraining initial layers, the model can leverage pre-trained representations while allowing downstream layers to adapt to the specific SQL translation task
Hyperparameters	Learning rate: 1e-4, batch size: 16, AdamW optimizer, cosine scheduler with warmup, 20 epochs, early stopping after 8 epochs

Table 3: Details of the best-performing T5 model configurations (fine-tuned)

Q6.

System	Query EM	F1 score
Dev Results		
T5 fine-tuned		
T5 finetuned (no layer freezing)	0.00	38.27
T5 finetuned (2 layer freezing)	0.032	66.24
Test Results		
T5 fine-tuning	0.00	60.82

Table 4: Development and test results

Quantitative Results:

Qualitative Error Analysis:

Error Type	Example Of Error	Error Description	Statistics
Underconstrained Query	NL: "flights from phoenix to milwaukee" SQL: Identical multi-city routing query	Queries that produce identical result sets despite different SQL structures. Model generates SQL that matches ground truth semantically but with slight structural variations, indicating a lack of precise query specificity and potential over-generalization in query generation.	172/466
Malformed SQL	NL: "show me the flights arriving in baltimore from philadelphia at about 4 o'clock" SQL: Incorrect arrival time and city order conditions	SQL queries with structural defects including incorrect join conditions, misplaced filtering criteria, and syntactical issues that fundamentally prevent accurate query execution and result retrieval, particularly in complex temporal and geographical filtering scenarios.	144/466
Logical Condition Error	NL: "what are the flights available between 10am and 3pm between pittsburgh and fort worth" SQL: Incomplete time range filtering	Systematic misinterpretation of temporal constraints, including incorrect time range implementations, wrongly placed time conditions, and partial understanding of complex time-based filtering requirements in multi-city flight searches.	98/466
Routing and Contextual Ambiguity	NL: "denver to atlanta" SQL: Partial routing implementation	Residual error category capturing miscellaneous query generation issues not classified in primary error types, including subtle routing complexities, partial condition implementations, and edge-case query transformations that deviate from expected semantic mapping.	28/466

Table 5: Qualitative Error Analysis on the Development Set

Q7.

Provide a link to a google drive which contains a model checkpoint used to generate outputs you have submitted.

https://drive.google.com/drive/folders/1ulHbQx7JQd0j2RWR5m-W1t4jURO-2tLk?usp=sharing

Extra Credit:

If you are doing extra credit assignment, please describe your system here, as well as provide a link to a google drive which contains a model checkpoint used to generate outputs you have submitted.

In implementing the T5 model training from scratch for the Text-to-SQL task, I carefully initialized the model with random weights while maintaining the architectural integrity of the T5 small configuration. The key implementation began in the `initialize_model()` function, where instead of loading pretrained weights, I used `T5Config.from_pretrained()` to create a configuration and then instantiated `T5ForConditionalGeneration` with these random weights.

To ensure compatibility, I resized the token embeddings to match our specific tokenizer, setting the correct padding, decoder start, and end token IDs. The training process involved using a lower learning rate of 5e-5, extended warmup epochs, and data augmentation strategies to help the model learn effectively.

These augmentations included: Synonym replacements, Structural variations in natural language queries and random word order shuffling

During training, I implemented a comprehensive logging mechanism to track the model's learning trajectory, monitoring metrics like training loss, record F1 score, and error rates. The model was trained for up to 30 epochs with an early stopping mechanism based on development set performance, allowing the randomly initialized model to gradually learn the complex task of translating natural language to SQL queries.

Performance Highlights

- Initial Record F1 Score: ≈ 0.12
- Final F1 score: 60.82%

https://drive.google.com/drive/folders/1bgHYa6qzOH2QZz3E8f6Y6kfeVyzrTSGl?usp=sharing