DSGA 1011: Assignment 4

Megh Rathod
Net ID: mr7375

Q0. 1.

Please provide a link to your github repository, which contains the code for both Part I and Part II.

`https://github.com/meghrathod/nlp-ass-5`

Q2. 1.

Describe your transformation of dataset.

Our transformation strategy creates out-of-distribution (OOD) evaluation data by applying multiple realistic text modifications that preserve semantic meaning and sentiment labels. The transformation uses a multi-stage approach combining synonym replacement, verb tense changes, contraction expansion/contraction, and controlled typographical errors.

Transformation Components

1. Synonym Replacement (Two-Tier Approach): We employ a two-tier synonym replacement strategy with part-of-speech (POS) based probabilities:

- **Domain-Specific Groups (Primary):** We maintain curated synonym groups for movie review terminology, including:
  - Movie/film terms: {`movie, film, picture, motion picture`}
  - Acting/performance: {`acting, performance, portrayal, playing`}
  - Story/narrative: {`story, narrative, tale, plot, screenplay, script`}
  - Quality adjectives: {`good, great, excellent, fine, fantastic, outstanding`} and {`bad, terrible, awful, poor, horrible`}
  - Common verbs/adverbs: {`watch, see`}, {`very, extremely, really, quite`}

  These groups are stored as sets and automatically build bidirectional lookup dictionaries, ensuring no redundancy and easy maintenance.
- **WordNet Fallback:** For words not in domain groups, we query WordNet synsets using NLTK's `wordnet.synsets()` with appropriate POS tags. We filter synonyms to ensure they are alphabetic, differ from the original word, and are not duplicates.
- **POS-Based Probabilities:** Replacement probabilities vary by word type to balance semantic preservation with transformation aggressiveness:
  - Adjectives: 80% (`p_synonym_adj = 0.80`)
  - Adverbs: 80% (`p_synonym_adv = 0.80`)
  - Nouns: 65% (`p_synonym_noun = 0.65`)
  - Verbs: 80% (`p_synonym_verb = 0.80`)

  Higher probabilities for adjectives/adverbs allow semantic variation while maintaining sentiment, while verbs and nouns are transformed more conservatively to preserve core meaning.

2. Verb Tense Changes: With probability 75% (`p_tense_change = 0.75`), we change verb tenses using a comprehensive mapping dictionary covering common verb forms (e.g., present ↔ past: `is` ↔ `was`, `goes` ↔ `went`, `watches` ↔ `watched`). This introduces temporal variation while preserving semantic content.

3. Contraction Expansion/Contraction: With probability 20% (`p_contraction = 0.20`), we randomly expand contractions (e.g., `don't` → `do not`) or contract expanded forms (e.g., `I am` → `I'm`). This simulates natural language variation in formality and style.

4. Controlled Typographical Errors: We introduce realistic typing errors using QWERTY keyboard proximity simulation:

- Probability: 40% per eligible word (`p_typo = 0.40`)
- Maximum: 10 typos per example (`max_typos_per_example = 10`)
- Method: For words longer than 3 characters, we randomly select a character position (excluding first and last) and replace it with a nearby QWERTY key. For example, `a` can be replaced by `s`, `q`, `w`, or `e`; `e` by `w`, `r`, `d`, or `s`.
- Constraint: Typos are only introduced if the word was not already transformed by synonym replacement, ensuring transformations don't compound.

Implementation Details

The transformation pipeline processes text as follows:

1. **Tokenization:** Use NLTK's `word_tokenize()` to preserve sentence structure
2. **POS Tagging:** Tag tokens with NLTK's `pos_tag()` to identify word types
3. **Transformation Stages:** Apply transformations in order: (1) contraction handling, (2) synonym replacement (domain groups then WordNet), (3) verb tense changes, (4) typo introduction
4. **Capitalization Preservation:** All replacements maintain the original word's capitalization pattern (uppercase, lowercase, title case)
5. **Detokenization:** Reconstruct sentences using `TreebankWordDetokenizer()` to restore natural spacing and punctuation

Why This Transformation is Reasonable

- **Realistic Variations:** Synonym usage, tense changes, contractions, and typos all occur naturally in user-generated content, especially in informal movie reviews.
- **Preserves Semantics:** All transformations maintain the core meaning and sentiment of the original text, ensuring labels remain valid.
- **Appropriate Complexity:** The transformation is non-trivial (requires semantic understanding) but not extreme (text remains readable and interpretable).
- **Test-Time Plausibility:** These variations are commonly observed in real-world test scenarios, making the OOD evaluation meaningful.

Example Transformation

Original: "Titanic is the best movie I have ever seen. The acting was excellent and the story was compelling."

Transformed: "Titanic was the finest film I have ever seen. The performance was outstanding and the narrative was compelling."

Changes: `is` → `was` (tense), `best` → `finest` (synonym), `movie` → `film` (domain group), `acting` → `performance` (domain group), `excellent` → `outstanding` (domain group), `story` → `narrative` (domain group).

Q3. 1

Report & Analysis

Accuracy Results

Model	Original Test Set	Transformed Test Set
Baseline (no augmentation)	93.57%	84.95%
With data augmentation	93.61%	90.13%

Table 1: Accuracy on original and transformed test sets.

Analysis and Discussion

(1) Model Performance on Transformed Test Data After Augmentation:

After applying data augmentation during training, the model's performance on the transformed test data **improved** from 84.95% to 90.13%, representing a gain of 5.18 percentage points. This suggests that the augmentation strategy successfully exposed the model to similar transformations during training, improving robustness to lexical variations, typos, and other surface-level changes while maintaining semantic understanding.

(2) Impact of Data Augmentation on Original Test Data:

Data augmentation **slightly enhanced** the model's performance on the original test set from 93.57% to 93.61%, a marginal improvement of 0.04 percentage points. This indicates that the augmented training data helped the model learn more generalizable features that also benefit performance on standard test data, without introducing significant distribution shift that would hurt performance on the original distribution.

Intuitive Explanation

The observed results can be explained by considering how data augmentation affects model training:

- **Improved OOD Robustness:** When the model is trained on augmented data that includes synonym replacements, tense changes, contractions, and typos, it learns to focus on semantic patterns rather than memorizing specific lexical choices. This makes it more robust to the lexical variations present in the transformed test set.
- **Regularization Effect:** The augmented training examples act as a form of regularization, preventing the model from overfitting to exact word sequences and encouraging it to learn more generalizable representations.
- **Trade-off with Original Distribution:** If augmentation is too aggressive or introduces patterns not present in the original distribution, it may hurt performance on the original test set. Conversely, if augmentation is well-calibrated, it can improve generalization to both original and transformed data.
- **Semantic Preservation:** Since our transformations preserve semantic meaning and sentiment, the model learns that different surface forms can express the same underlying sentiment, improving its ability to handle natural language variation.

Limitation of the Data Augmentation Approach

One key limitation of this data augmentation approach for improving OOD performance is that it only addresses **lexical and surface-level variations** (synonyms, typos, contractions, tense changes) but does not handle **semantic or structural distribution shifts**. For example:

- The augmentation cannot help with domain shifts where the underlying sentiment indicators change (e.g., reviews from a different genre or cultural context where "dark" might be positive rather than negative).

- It does not address cases where the transformation introduces subtle semantic shifts that our heuristics fail to detect, potentially creating label noise.
- The approach assumes that synonym replacement and other transformations perfectly preserve sentiment, which may not always hold true in practice (e.g., “terrible” vs. “poor” may have different intensity levels).
- The augmentation is limited to transformations we can programmatically define, missing more complex linguistic variations like paraphrasing, style changes, or discourse-level modifications that occur in real OOD scenarios.

To truly improve OOD robustness, we would need to combine lexical augmentation with other techniques such as adversarial training, domain adaptation, or learning from diverse data sources that capture broader distribution shifts.

Part II. Q4

Statistics Name	Train	Dev
Number of examples	4225	466
Mean sentence length	10.96	10.91
Mean SQL query length	60.90	58.90
Vocabulary size (natural language)	868	444
Vocabulary size (SQL)	644	393

Table 2: Data statistics before any pre-processing.

Statistics Name	Train	Dev
T5 fine-tuned model (T5TokenizerFast)		
Mean NL input length (tokens, no prompt)	18.10	18.07
Mean NL input length (tokens, with prompt)	23.10	23.07
Mean SQL query length (tokens)	217.37	211.05
Vocabulary size (natural language, token IDs)	796	470
Vocabulary size (SQL, token IDs)	556	396

Table 3: Data statistics after T5 tokenization. The prompt “translate English to SQL: ” adds 5 tokens to each input. Token statistics include special tokens (BOS/EOS).

Q5

Design choice	Description
Data processing	Natural language questions and SQL queries are loaded without additional preprocessing. The database schema is provided in the input prompt to help the model understand table/column relationships. No data augmentation was applied to the training set.
Tokenization	We use the default T5TokenizerFast from HuggingFace (`google-t5/t5-small`). Encoder input format: "translate English to SQL: [natural language question]". Decoder receives teacher-forced SQL tokens during training. Maximum sequence lengths: 512 tokens for both encoder and decoder.
Architecture	Fine-tuned the entire T5-small model (60M parameters) including both encoder and decoder layers. No layers were frozen. Gradient checkpointing was available but not used in final model (sufficient memory with batch size 32). Mixed precision (FP16) training was enabled for 2x speedup.
Hyperparameters	**Phase 1 (Cold Start, Epochs 0-19):** LR=5e-5, batch size=16, gradient accumulation=2 (effective batch=32), cosine scheduler with 2 warmup epochs, weight decay=0.01, patience=5 epochs. Achieved 36.66% F1. **Phase 2 (Ultra-Low LR Fine-tuning): LR=2e-6** (constant, 25x lower), batch size=48, gradient accumulation=1 (effective batch=48), weight decay=0.0001, patience=1 epoch. This ultra-low LR was critical for staying near the optimal checkpoint while making microscopic improvements. Achieved 63.93% F1. **Optimization:** AdamW optimizer, mixed precision (FP16), eval beam size=5. Hardware: L4 GPU (24GB VRAM).

Table 4: Details of the best-performing T5 model configurations (fine-tuned)

Key Training Insights and Design Decisions

1. Critical Discovery - Checkpoint Proximity and Ultra-Low LR: When resuming from a well-trained checkpoint, we discovered the model was already near-optimal. Multiple experiments revealed:

- **LR=5e-5 with warmup:** 53.1% → 41.1% F1 (overshoot and crash)
- **LR=1e-5 constant:** 58.6% → 38.7% F1 (still too high, moved away from optimum)
- **LR=2e-6 constant:** 63.2% → 63.9% F1 (success! Microscopic adjustments preserved optimum)

Key Insight: The checkpoint was in a sharp, high-quality local optimum. Ultra-low LR (2e-6, 25x lower than initial training) allowed the model to explore the immediate neighborhood without falling into worse regions.

2. Memory and Speed Optimizations:

- **Mixed Precision (FP16):** Enabled 2x speedup and reduced memory from 12GB to 6GB per batch
- **Batch Size Scaling:** Started with batch size 16 (effective 32 with grad accumulation) in Phase 1, scaled to 48 in Phase 2 when memory allowed

- **Fast Evaluation Strategy:** Full evaluation with beam search generation takes 3 minutes per epoch. Implemented "fast eval" (loss-only, no generation) for non-checkpoint epochs, reducing total eval time by 60%
- **Gradient Accumulation:** Used in Phase 1 (accumulation steps=2) to achieve effective batch size of 32; Phase 2 used direct batch size of 48

3. Two-Phase Training Strategy:

- **Phase 1 (Cold Start):** Higher LR (5e-5), cosine scheduler with warmup, aggressive weight decay (0.01) → Achieved 36.66% F1 by epoch 14
- **Phase 2 (Ultra-Low LR Fine-tuning):** Ultra-low constant LR (2e-6, 25x lower than Phase 1), no scheduler, reduced weight decay (0.0001), higher patience → Prevents overshoot, enables stable improvement from good checkpoint, achieved 63.93% F1

4. Hardware Considerations: On L4 GPU (24GB VRAM): 1.5 min/epoch with optimizations (mixed precision, batch size 48). The L4's higher memory capacity compared to T4 allowed for larger batch sizes, contributing to faster training throughput.

Q6.

System	Record EM (%)	Record F1 (%)
Dev Results		
T5 fine-tuned (Phase 1: Epochs 0-19)		
Best checkpoint (Epoch 14)	29.61	36.66
End of Phase 1 (Epoch 19)	23.39	27.72
T5 fine-tuned (Phase 2: Ultra-Low LR)		
Epoch 0 (LR=2e-6)	55.79	63.16
Epoch 1	56.44	63.87
Epoch 2	56.44	63.93
Best Model (to be evaluated on test set)		
T5 fine-tuning (Phase 2, Epoch 2)	56.44	63.93

Table 5: Development set results showing training progression. Phase 1: LR=5e-5 with cosine scheduling achieved 36.66% F1. Phase 2: Ultra-low constant LR=2e-6 (critical discovery) achieved 63.93% F1 by staying near the optimal checkpoint. Best model (Epoch 2) shows Record EM of 56.44% and Record F1 of 63.93% on dev set. Test set evaluation pending. Note: Record-based metrics focus on database query result accuracy rather than SQL syntax matching.

Quantitative Results:

Training Progress Notes: Initial training (Phase 1) achieved steady improvement from 0% to 36.66% F1 over 14 epochs, then began overfitting. Multiple attempts to resume training revealed a critical insight: the checkpoint was already near-optimal, and any significant learning rate caused degradation.

Key Discovery: Using ultra-low LR (2e-6, 25x lower than Phase 1) allowed the model to make microscopic adjustments while staying in the optimal region, achieving 63.93% F1 with stable improvements. This demonstrates that the Phase 1 checkpoint had learned excellent representations but needed extremely gentle fine-tuning to unlock its full potential.

Qualitative Error Analysis:

Error Type	Example Of Error	Error Description	Statistics
Missing Temporal Constraints	**Query:** "flights available tomorrow from Denver to Philadelphia" **Generated:** Missing date/day joins (no `days_1`, `date_day_1` tables)	Model fails to capture temporal references like "tomorrow", "Thursday", "morning". The generated SQL omits necessary joins to `days` and `date_day` tables to filter by specific dates/times.	High prevalence: ∼30-40% of queries with temporal references (estimated 150/466 dev examples)
Missing Time-of-Day Filters	**Query:** "afternoon flights from Washington to Boston" **Generated:** Missing `departure_time BETWEEN 1200 AND 1800` clause	Model fails to translate time-of-day references ("morning", "afternoon", "before 9am") into appropriate `departure_time` or `arrival_time` range constraints.	Medium prevalence: ∼20% of queries with time-of-day (estimated 90/466)
Incorrect Connecting Flight Logic	**Query:** "fly from Philadelphia to San Francisco through Dallas" **Generated:** Uses duplicate city constraints instead of `flight_stop` table join	Model struggles with multi-leg journeys requiring `flight_stop` table. Instead of joining on `flight_stop.stop_airport`, it incorrectly adds redundant destination city conditions.	Medium-high: ∼25% of connecting flight queries (estimated 40/160)
Over-simplified Schema	Multiple examples generate basic city-to-city queries without constraints	Model often produces minimal viable queries that match schema structure but miss crucial filtering conditions (fares, times, specific airlines, aircraft types). This suggests difficulty handling complex multi-table joins and predicate composition.	Very high: ∼50-60% of all queries (280/466) lack at least one constraint present in ground truth

Table 6: Qualitative error analysis on dev set. Errors predominantly stem from incomplete semantic parsing of temporal/spatial constraints and difficulty with complex multi-table join patterns. The model successfully learns basic schema structure (city/airport joins) but struggles with compositional reasoning required for full query specification.

Q7.

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

`https://drive.google.com/file/d/1q8lVyqCE7kEfqCZLsR19WJ-sOQD62IXa/view?usp=sharing`

Model Details: T5-small fine-tuned with ultra-low learning rate (2e-6) strategy, achieving 63.93% Record F1 and 56.44% Record EM on development set. The model was trained in two phases: Phase 1 (cold start, LR=5e-5) reached 36.66% F1, then Phase 2 (ultra-low LR=2e-6) unlocked the checkpoint's full potential by making microscopic weight adjustments while staying in the optimal region.

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. Optional TODO