**Technical Report: NLP Inference Pipeline**

**Data Handling**

* **Synthetic Dataset Creation**:
  + A synthetic dataset (calls\_dataset.csv) with 150 entries was generated, representing sales/marketing call snippets. Each snippet is associated with 1–3 labels out of four categories:
    - **Competition, Pricing Discussion, Security, Objection**.
  + Example: id, text\_snippet, labels

1, "We love the analytics, but CompetitorX has a cheaper subscription.", "Pricing Discussion, Objection, Competition"

* **Domain Knowledge Base**:
  + A dictionary (domain\_knowledge.json) was created for entity extraction, with the following structure:

{

"competitors": ["CompetitorX", "CompetitorY", "CompetitorZ"],

"features": ["analytics", "AI engine", "data pipeline"],

"pricing\_keywords": ["discount", "renewal cost", "budget", "pricing model"]

}

* **Preprocessing**:
  + Text snippets were cleaned to remove:
    - Stopwords.
    - Special characters.
    - Case inconsistencies.
* **Data Augmentation**:
  + Snippets with minority labels were augmented by duplicating and modifying examples, balancing the dataset.
  + Example :  
    Original: "CompetitorY provides discounts for first-time customers." Augmented: "CompetitorY offers discounts for early adopters."

**Modeling Choices**

 **Approach**:

* **Model**: Logistic Regression (One-vs-Rest strategy).
  + **Why Logistic Regression?**
    - Logistic Regression is a simple yet effective model for multi-label classification problems, especially when the dataset is moderately sized.
    - It handles binary classification well, and the One-vs-Rest strategy extends its functionality to multi-label classification.
    - It is interpretable, making it easier to understand how different features contribute to label predictions.
    - Computational efficiency allows for fast training and prediction compared to more complex models like neural networks.
  + **Why One-vs-Rest?**
    - In multi-label classification, each label is treated as an independent binary classification problem.
    - This simplifies the problem and ensures each label gets dedicated attention during training, which helps when labels are not mutually exclusive (e.g., "Pricing Discussion" and "Objection" can co-occur).
* **Feature Extraction**: TF-IDF Vectorizer with max\_features=5000.
  + **Why TF-IDF?**
    - TF-IDF (Term Frequency-Inverse Document Frequency) is a well-established technique for text vectorization.
    - It assigns importance to terms that are frequent in a document but rare across the entire dataset, which helps the model focus on distinguishing features.
    - The max\_features=5000 ensures that only the top 5000 terms are considered, balancing dimensionality and feature richness.
    - It’s interpretable, meaning we can analyze which terms have the most significant impact on predictions.

 **Hyperparameter Tuning**:

* **Why Tune Regularization Strength (C)?**
  + C controls the trade-off between model complexity and regularization.
  + A lower C value enforces stronger regularization, reducing overfitting but potentially underfitting the data.
  + A higher C allows the model to fit the training data better but may lead to overfitting.
  + Grid search ensures that we systematically test multiple values to find the optimal balance.
* **Why l2 Penalty?**
  + The l2 penalty adds regularization by penalizing large coefficients, encouraging the model to focus on meaningful features and avoid overfitting.
  + It’s particularly effective in high-dimensional spaces like those produced by TF-IDF vectorization.

 **Why This Approach Works**:

* Logistic Regression with TF-IDF is a proven pipeline for text classification tasks.
* It balances interpretability and performance, providing actionable insights into why a snippet was classified into specific categories.
* Hyperparameter tuning ensures optimal performance tailored to the dataset.

**3. Performance Results**

**Classification**

* **Cross-Validation Scores**: [0.7850, 0.8205, 0.7333, 0.7544, 0.7843]
* **Mean F1-Score**: 0.7755
* **Standard Deviation**: 0.0297
* **Classification Report**:

precision recall f1-score support

Competition 0.64 0.82 0.72 11

Objection 0.53 0.67 0.59 12

Pricing Discussion 0.69 1.00 0.82 18

Security 1.00 1.00 1.00 18

micro avg 0.73 0.90 0.80 59

macro avg 0.72 0.87 0.78 59

weighted avg 0.74 0.90 0.81 59

samples avg 0.79 0.91 0.82 59

**Entity Extraction**

* **Dictionary Lookup**:
  + **What It Does**:
    - Texts are compared against a predefined dictionary (domain\_knowledge.json) containing domain-specific entities like competitors, features, and pricing terms.
  + **Why Chosen?**
    - Directly mapping known terms ensures high precision for domain-specific entities.
    - It is computationally efficient and works well for static or semi-static vocabularies, like "CompetitorX" or "analytics."
    - Ensures that critical terms from the domain are always captured.
* **Regex-Based Extraction**:
  + **What It Does**:
    - Uses regular expressions (regex) to match patterns like competitor names (r"\bCompetitor[A-Z]\b") or pricing terms (r"\b(?:discount|renewal cost|pricing model)\b").
  + **Why Chosen?**
    - Regex provides flexibility to capture entities beyond the static dictionary, such as variations in text that aren’t explicitly listed.
    - It handles dynamic and structured patterns, such as competitors with similar naming conventions ("CompetitorX," "CompetitorY").
    - Regex can quickly adapt to new patterns without retraining a model.
* **Combined Approach**:
  + **What It Does**:
    - Merges results from dictionary lookup and regex to produce a comprehensive set of entities.
    - Example: A snippet like "CompetitorX offers analytics and discounts" would result in:
      * From Dictionary: [{type: "competitors", value: "CompetitorX"}, {type: "features", value: "analytics"}]
      * From Regex: [{type: "competitor", value: "CompetitorX"}, {type: "pricing", value: "discount"}]
  + **Why Combine?**
    - Dictionary lookup ensures domain-specific entities are captured with high precision.
    - Regex extends coverage to patterns not explicitly listed in the dictionary.
    - The combination increases recall while maintaining precision, which is crucial for real-world applications where both known and unknown entities are present.
* **Challenges and Solutions**:
  + **Challenge**: Dictionary lookup alone misses terms outside the predefined list.
    - **Solution**: Introduced regex for broader coverage.
  + **Challenge**: Regex alone can overfit to patterns and produce false positives.
    - **Solution**: Merged dictionary results with regex to ensure only relevant entities are captured.
* **Why This Approach Works**:
  + Combining dictionary lookup and regex creates a robust system that captures both static and dynamic entities.
  + It’s flexible and can scale to new terms or patterns with minimal adjustments.
  + The system balances precision and recall, ensuring critical entities are not missed while avoiding overextraction.
* **Results**:
  + Dictionary-based extraction:  
    [{type: "competitors", value: "CompetitorX"}, {type: "features", value: "analytics"}]
  + Combined extraction:  
    [{type: "competitors", value: "CompetitorX"}, {type: "features", value: "analytics"}, {type: "competitor", value: "CompetitorX"}, {type: "feature", value: "analytics"}]
* **Evaluation**:
  + **Precision**: 0.50
  + **Recall**: 1.00

**4. Error Analysis**

* + Confusion between "Objection" and "Pricing Discussion" due to overlapping keywords like "renewal cost."
  + Snippets with vague wording led to misclassifications.
  + Example:
    - Text: "Our budget is tight; can we negotiate pricing?"
    - Predicted: ["Pricing Discussion"]
    - Actual: ["Pricing Discussion", "Objection"].
  + Example:
    - **Text**: "Can we get a discount on the renewal fee next year?"  
      **Extracted Entities**: ["pricing", "discount"]  
      **Missed Entity:** "renewal fee" (a variant of "renewal cost")
* **Entity Extraction**:
  + Dictionary-based methods failed to capture unexpected terms or abbreviations.
  + Regex missed entities with complex wording or typos.

**Areas of Improvement:** Improve the model’s ability to distinguish between labels with similar keywords, possibly by integrating context-aware models or adjusting thresholds for label assignment.

**5. Future Work**

* Create a bigger and better dataset to make it more diverse and for better results
* Use Transformer models for classification and entity extraction