**Technical Report on Multi-Label Text Classification and Entity Extraction**

**1. Data Handling**

**1.1 Preprocessing:**

* **Text Cleaning:** Text snippets were converted to lowercase, and non-alphanumeric characters were removed using regular expressions.
* **Stopword Removal:** Common English stopwords were filtered out using the NLTK stopword corpus.
* **Lemmatization:** Words were reduced to their base forms using NLTK’s WordNet Lemmatizer to standardize the vocabulary.

**1.2 Label Encoding:**

* Labels were stored as comma-separated strings. These were converted to a multi-hot encoding using MultiLabelBinarizer for compatibility with multi-label classification models.

**1.3 Data Splitting:**

* The dataset was split into 80% training and 20% testing using train\_test\_split from scikit-learn, ensuring a balanced representation of labels.

**1.4 Data Augmentation:**

* To handle label imbalance, random oversampling was applied. Minority label samples were duplicated using the resample method from scikit-learn.

**2. Modeling Choices**

**2.1 Classification Model:**

* **TF-IDF Vectorization:** Text data was transformed into numerical feature vectors using TF-IDF with a maximum of 5000 features.
* **Logistic Regression with One-vs-Rest Strategy:** Logistic regression was selected due to its simplicity and interpretability, combined with the One-vs-Rest (OvR) framework to handle multiple labels.

**2.2 Hyperparameter Tuning:**

* A grid search with 3-fold cross-validation was conducted to optimize hyperparameters for Logistic Regression, including:
  + Regularization strength (C): [0.1, 1, 10].
  + Penalty terms (l1, l2).
  + Solvers (liblinear).

**2.3 Entity Extraction Model:**

* **Dictionary-Based Lookup:** Predefined domain knowledge (competitors, features, pricing keywords) was matched against the text using direct string matching.
* **spaCy PhraseMatcher:** A spaCy pipeline was augmented with a custom PhraseMatcher to identify domain-specific entities based on labeled phrases.

**Challenges and Solutions:**

* **Challenge:** Significant class imbalance in the dataset led to poor recall for minority labels.
  + **Solution:** Applied oversampling of minority classes to ensure balanced training data.
* **Challenge:** Ambiguity in text snippets caused misclassification and overlapping predictions.
  + **Solution:** Experimented with different vectorization techniques and refined preprocessing to minimize noise.

**3. Performance Results**

**3.1 Multi-Label Classification Metrics:**

* After hyperparameter tuning, the following metrics were achieved on the test set:
  + **Hamming Loss:** 0.12
  + **Macro F1-Score:** 0.58
  + **Jaccard Similarity (Macro):** 0.61

**3.2 Entity Extraction Results:**

* Performance was partially evaluated using manually labeled test examples:
  + **Precision:** 0.82
  + **Recall:** 0.75
  + **F1-Score:** 0.78

**4. Error Analysis**

**4.1 Misclassification Examples:**

* Example 1: “CompetitorX offers better pricing than your product” was misclassified under the Objection label instead of Competition.
  + **Reason:** Semantic overlap between labels.
* Example 2: “Can you lower the subscription price?” was classified under both Pricing Discussion and Objection.
  + **Reason:** Context ambiguity.

**4.2 Confusion Matrix Observations:**

* Overlapping predictions were common for Objection and Pricing Discussion.
* Minority labels like Positive had poor recall due to insufficient training samples.

**4.3 Areas for Improvement:**

* Better disambiguation techniques for semantically similar labels.
* Advanced text embeddings (e.g., BERT) for richer contextual understanding.

**5. Future Work**

**5.1 Data Curation:**

* Collect more balanced datasets, particularly for underrepresented labels like Positive.
* Label refinement to reduce ambiguity and overlap.

**5.2 Advanced Modeling:**

* Fine-tune pre-trained transformer models (e.g., BERT, RoBERTa) for multi-label classification.
* Experiment with hierarchical attention networks to better handle label dependencies.

**5.3 Enhanced Entity Extraction:**

* Incorporate contextual embeddings into the entity extraction pipeline.
* Expand domain knowledge dictionaries with synonyms and variations for more comprehensive coverage.

**5.4 Real-Time Deployment:**

* Implement a feedback loop to refine model performance over time based on user corrections.
* Integrate the model into an end-to-end pipeline for real-time classification and entity extraction.

This report summarizes the steps, challenges, and results of building a multi-label classification and entity extraction system. Future work aims to address current limitations and enhance the model's robustness for production environments.