BERT & ERNIE

Project for the MNLP University Course

Andrea Gravili
Sapienza University of Rome
gravili.2180997@studenti.uniroma1.it

Olmo Ceriotti
Sapienza University of Rome
ceriotti.2193258@studenti.uniroma1.it

1 Introduction

This project classifies cultural items as Agnostic, Representative, or Exclusive using two approaches: a fine-tuned SBERT and a XGBoost classifier.

2 Methodology

This section outlines two approaches for classifying cultural items: an LLM-based method using Sentence-BERT (`SBERT`) and `SVM`, and a feature-engineering method using `XGBoost`.

2.1 LLM-Based Approach

This approach leverages `SBERT` fine-tuned with Triplet Loss for generating meaningful embeddings, followed by a Support Vector Machine (`SVM`) for classification.

Data Augmentation: Recognizing the limited data per item, we augmented the dataset by retrieving the full text content from the corresponding Wikipedia page (linked within the original dataset) and appending this text to each input data point.

Text Embedding with SBERT: We employed `SBERT`, specifically the pre-trained `"all-mpnet-base-v2"` model, to transform the augmented text into fixed-size numerical embeddings. Unlike token-level transformers like `BERT`, `SBERT` is designed (often via pooling strategies) to generate vector representations for e ntire sentences or paragraphs, making it suitable for semantic comparison tasks.

Fine-tuning with Triplet Loss: To adapt the pre-trained model to our cultural classification task, we fine-tuned it using Triplet Loss. This loss function operates on triplets of data points: an **Anchor (A)**, a **Positive (P)** from the same cultural class, and a **Negative (N)** from a different class. The objective is to minimize the distance between Anchor and Positive embeddings ($d(A, P)$) while maximizing the distance between Anchor and Negative embeddings ($d(A, N)$) by at least a predefined margin α. The loss for a single triplet $(A^{(i)}, P^{(i)}, N^{(i)})$ is:

$$L^{(i)} = \max\left(\left\|f(A^{(i)}) - f(P^{(i)})\right\|_2^2 - \left\|f(A^{(i)}) - f(N^{(i)})\right\|_2^2 + \alpha,\ 0\right) \tag{1}$$

where $f(\cdot)$ is the `SBERT` embedding function and $\|\cdot\|_2^2$ denotes the squared Euclidean distance. The fine-tuning process aims to map texts of the same cultural category to nearby points in the embedding space, while separating texts from different categories.

Final Classification: After fine-tuning, the `SBERT` model serves as a feature extractor. We tested different models for the classification task. In the end, an `SVM` classifier was then chosen to perform the classification task.

2.2 Non-LLM Based Approach

This approach focuses on extracting explicit features from text and metadata, combining them, and using an `XGBoost` classifier, thus avoiding large transformer models.

Wikidata Feature Extraction: Utilizing the Wikidata links provided in the dataset, we extracted structured information for each item:

- **Sitelink Count:** The number of links from other Wikimedia projects, acting as a proxy for global recognition.
- **Country of Origin (`P495`):** Presence indicates specific geographic association.
- **Inception/Publication Date (`P571`/`P577`):** Used to estimate the item's age.

These features provide valuable non-textual context.

Lexical Feature Extraction: We defined keyword lexicons (`nationality`, `cultural`, `global`, `representative`) targeting relevant semantic categories. The presence and frequency of these keywords within the item's text, identified using regular expressions, were used as features.

Text Representation (TF-IDF): We employed Term Frequency-Inverse Document Frequency (`TF-IDF`) using `scikit-learn`'s `TfidfVectorizer` to represent textual content numerically. This weights terms based on their frequency within a document (`TF`) and their rarity across the entire corpus (`IDF`), down-weighting common words.

$$\text{TF}(t, d) = \frac{\text{freq}(t, d)}{|d|} \quad \text{IDF}(t, D) = \log \frac{|D|}{1 + \text{df}(t)} + 1$$

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

Combined Text Features (NER + Lexicon): We generated features by combining Named Entity Recognition (`NER`) performed with the `spaCy` library (identifying entities like locations, organizations) with the lexical features. Features included counts of relevant entity types and lexicon keyword matches.

Feature Integration: All extracted features (`TF-IDF` vectors, `NER` counts, Lexicon counts/flags, `Wikidata` features) were concatenated into a single feature vector for each data item. Since `TF-IDF`

yields sparse matrices, `scipy.sparse.hstack` was used to efficiently combine these with the other (potentially densified) feature sets into one wide sparse matrix.

Final Classification: An `XGBoost` classifier was trained on the resulting hybrid feature matrix. `XGBoost` is an efficient implementation of gradient boosted decision trees, which builds trees sequentially, with each new tree attempting to correct the errors made by the ensemble of preceding trees. We have also tried using only TF-IDF and only NER+Lexicon, but the hybrid was the best one.

3 Experiment

This section outlines the specific experimental setup used to implement and evaluate the described methodologies. We used Google Colab Pro with an A100 GPU, T4 GPU and M4 Pro Macbook.

3.1 LLM-Based Approach Experiment

- **Dataset:** The primary dataset was augmented by appending full Wikipedia text corresponding to each item's Wikidata link.
- **SBERT Model:** The pre-trained '"all-mpnet-base-v2"' SBERT model was used as the base.
- **Fine-tuning:** The SBERT model was fine-tuned using the Triplet Loss function (Eq. **??**) on the augmented training dataset.
- **Feature Extraction:** The fine-tuned SBERT model was used to generate embeddings (vectors) for all items in the training and validation sets.
- **SVM Training:** An SVM classifier was trained on the generated embeddings from the training set. Hyperparameter tuning for the SVM was performed using `RandomizedSearchCV`.
- **Evaluation:** The trained SVM was evaluated on the embeddings generated for the validation set using accuracy as the primary metric.

3.2 Non-LLM Based Approach Experiment

- **Dataset:** The original dataset including text fields and Wikidata links was used.
- **Feature Extraction Tools:**
  - `TfidfVectorizer` (from scikit-learn) was used to compute TF-IDF features from the combined text fields.
  - SpaCy (`en_core_web_sm` model) was used for Named Entity Recognition (NER) feature extraction.
  - Custom Python code with regular expressions was used for Lexicon keyword matching.
  - The `wikidatalibrary` was used to retrieve Wikidata features, with caching implemented.
- **Feature Processing:** Wikidata features were imputed (e.g., using median for missing age) and scaled (e.g., using `StandardScaler`). All features were combined into a single sparse matrix using `scipy.sparse.hstack`.
- **XGBoost Training:** An XGBoost classifier was trained on the resulting hybrid feature matrix derived from the training set. Hyperparameter tuning was performed for XGBoost.
- **Evaluation:** The performance of the tuned XGBoost model was assessed on the feature matrix derived from the validation set using accuracy as the primary metric.

4 Results

4.1 LLM-Based results

The SVM classifier, trained on fine-tuned SBERT embeddings and optimized via hyperparameter tuning, achieved an **accuracy of 0.78** on the validation set. Detailed performance visualizations, comparing the model before and after tuning, are presented in the Appendix. These include confusion matrices (Figures 1, 5), precision-recall curves (Figures 2, 6), ROC curves (Figures 3, 7), score distributions (Figures 4, 8), and the final predicted label distribution (Figure 9).

4.2 Non-LLM Based results

The XGBoost model, trained on the engineered hybrid feature set, achieved an **accuracy of 0.79** on the validation set. Visualizations detailing its performance and comparison with other models are provided in the Appendix. These include a model comparison overview (Figure 10), the XGBoost confusion matrix (Figure 11), its ROC curve (Figure 12), a training progression plot (Figure 13), and the distribution of predictions on unlabeled data (Figure 14).

5 Appendix

This appendix contains visualizations detailing the performance of the SVM classifier used in the LLM-based approach, both before (Initial) and after (Tuned) hyperparameter optimization.

5.1 Other experiments

Across the path that lead us to find the two proposed methods, we explored other techniques to complete the task in hand. For example:

- Fine-tuning for sequence classification of various BERT-based models.
- Fine-tuning for sequence classification of models of the ERNIE family
- Few-shot learning with retrieval augmented classification using SBERT and variety of LLMs.
- P-Tuning with a wide variety of LLMs.
- An array of ensemble techniques combining all of the above.

These experiments,s although interesting, didn't lead to any meaningful results, with a performance cap that we surpassed, using the techniques that we presented. It is worth noticing that the performance cap that we experienced could be due to computation limitations that restricted the size of our LLMs too much.

5.2 Initial SVM Performance

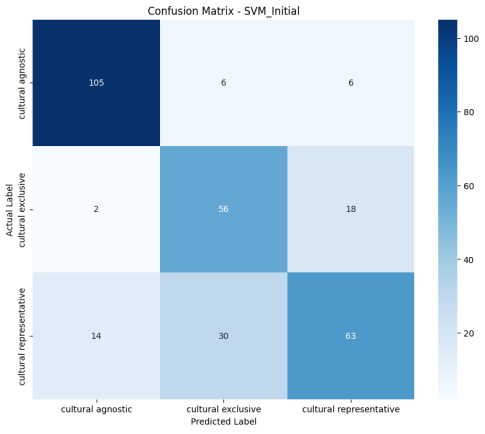

Figure 1: Confusion Matrix for the initial SVM model (before tuning).

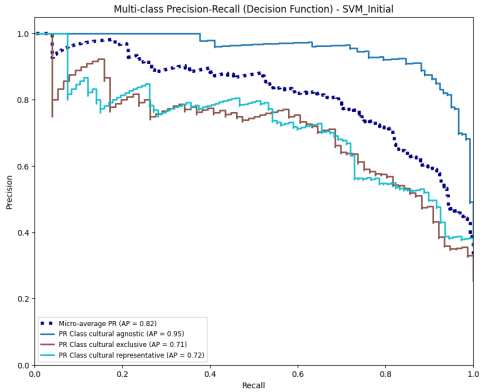

Figure 2: Precision-Recall Curve for the initial SVM model.

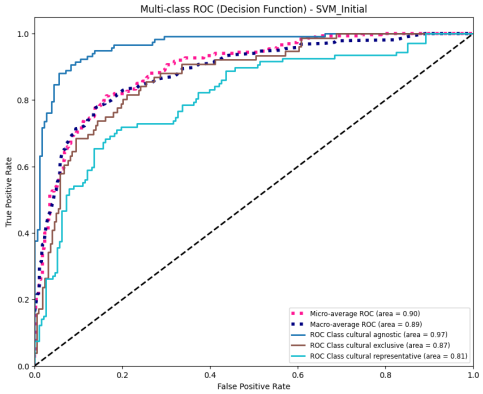

Figure 3: ROC Curve for the initial SVM model.

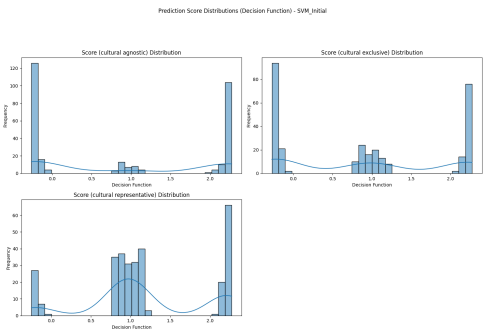

Figure 4: Score Distribution for the initial SVM model.

5.3 Tuned SVM Performance

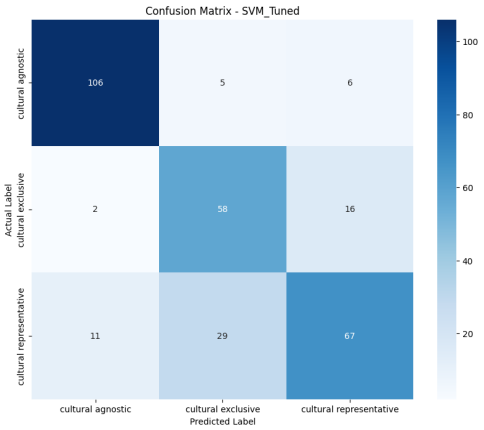

Figure 5: Confusion Matrix for the tuned SVM model.

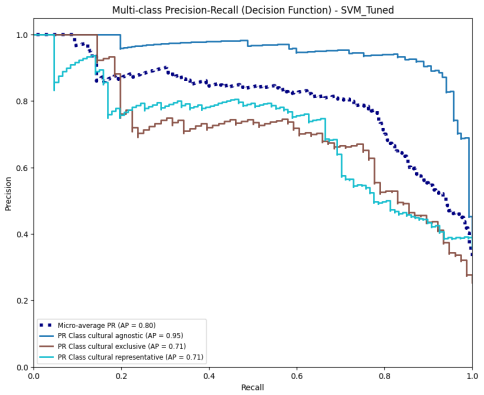

Figure 6: Precision-Recall Curve for the tuned SVM model.

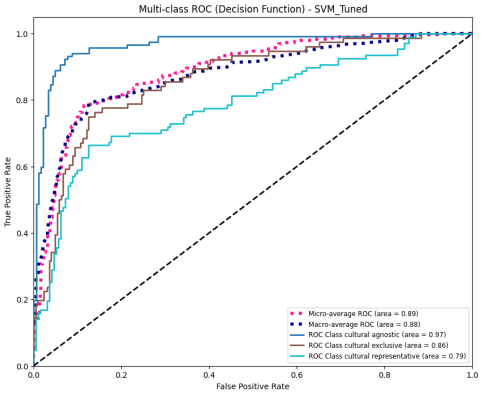

Figure 7: ROC Curve for the tuned SVM model.

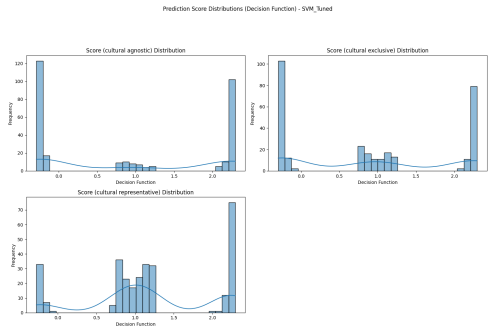

Figure 8: Score Distribution for the tuned SVM model.

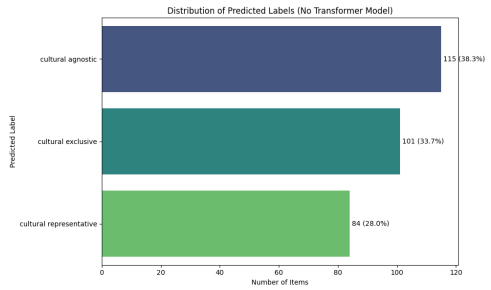

Figure 9: Distribution of Predicted Labels for the Tuned SVM Model.

5.4 XGBoost Performance and Model Comparison Visualization

This appendix contains visualizations related to the performance of the XGBoost classifier (Non-LLM approach) and model comparisons.

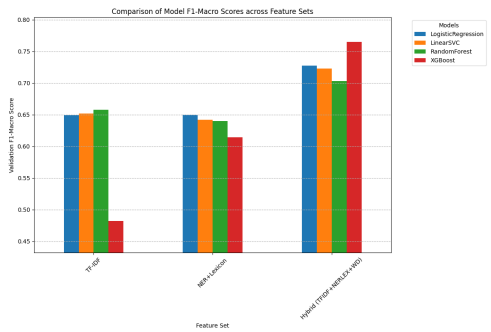

Figure 10: Comparison of Different Models Evaluated.

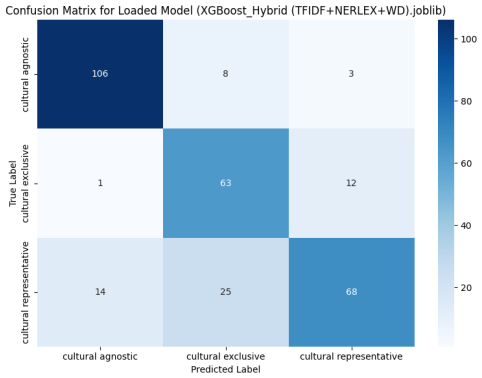

Figure 11: Confusion Matrix for the Tuned XGBoost Model.

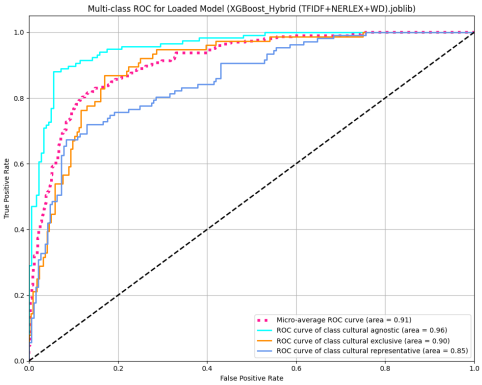

Figure 12: ROC Curve for the Tuned XGBoost Model.

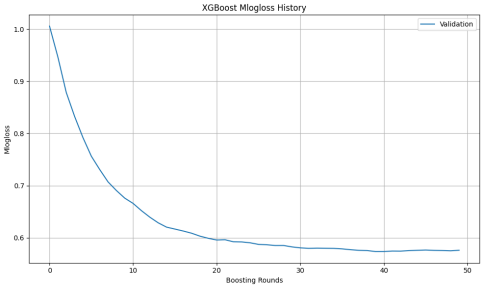

Figure 13: XGBoost Model Training Progression Plot.

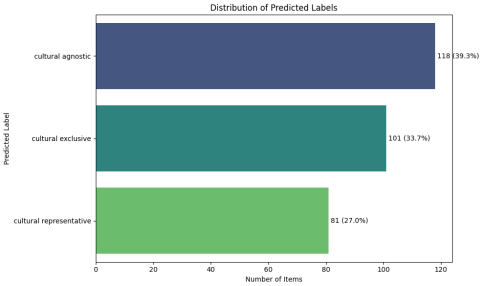

Figure 14: Distribution of Predicted Labels by XGBoost on Unlabeled Data.