Homework 1 - Cultural classification

A. Infantino 1922069 **A. Di Chiara 1938462** **F. Fragale 2169937**

1 Introduction

The aim of this project is to classify entities into three cultural categories: *culturally agnostic*, *culturally representative*, and *culturally exclusive*. To tackle this task, we first explored a non-language-model approach that combines embedding representations and numerical features extracted from Wikidata and Wikipedia. Then we used a language-model-based approach and compared its performance with the non-LM method.

2 Methodology

2.1 Dataset Enrichment

To improve the performance of our classification task, we enriched the dataset by extracting additional information from both Wikidata (wikidata.client) and Wikipedia (web scraping and wikipedia API). This enrichment process was guided by the observation of consistent patterns of wikidata/wikipedia pages of different items (e.g. the number of total views of a representative item is higher than the number of total views of an exclusive item). The data retrieved are summarized in Tab. 1.

2.2 Non-LM Approach

To classify items without using large language models, we designed a two-step method based on word embeddings and numerical features.
First, we cleaned the text by removing stopwords with the *nltk* library and then tokenized the text. We used the pre-trained *SpaCy* model *en_core_web_lg* to generate word embeddings for each item.
After embedding the items, we merged the *representative* and *exclusive* classes into a single class called *cultural*. Then, we computed two reference vectors: one for the *cultural* class and one for the *agnostic* class, by averaging the embeddings of the items in each class. To classify a new item, we calculated the cosine similarity between its embedding and the two reference vectors, and we assigned the label of the most similar one. Although the cosine similarity between the two reference vectors was high (0.9379), we found that some embedding dimensions—like 17, 105, and 268—showed strong differences (Fig. 1). This means that there were still some useful differences for classification.

To better separate the *representative* and *exclusive* items within the *cultural* class, we added a second classification step using numerical features. Each feature was given a weight based on their discriminative power: 0.6 for *num_languages*, 0.8 for *average_daily_views*, 0.1 for *word_count_en*, and 0.5 for *num_properties*. We calculated a score for each item using a weighted sum of its numerical features. Then, we computed the average score for the *representative* and *exclusive* items, called $\text{score}_{\text{repr}}$ and $\text{score}_{\text{excl}}$.
The threshold to separate the two classes was defined as:

$$\text{Thr} = \text{score}_{\text{excl}} + 0.1 \cdot (\text{score}_{\text{repr}} - \text{score}_{\text{excl}}) \quad (1)$$

The factor 0.1 helps to biases the correct classification of *representative* items, which were the most difficult to detect in earlier tests. If an item's score is below the threshold, it is classified as *exclusive*; otherwise, it is *representative*.

This final method reached an **accuracy of 0.76** and a **macro-averaged F1-score of 0.76** (Tab.2) (Fig.2).

2.3 LM approach

For the language model-based approach, we used RoBERTa with a custom classification head. First we prepared the dataset using special tags to help the model better understand the different parts of

the input.
The standard implementation from Hugging Face uses a simple linear layer on top of the [CLS] token, but we believed that adding a slightly deeper head could improve the model's performance.
Therefore, we kept the original logic of the forward pass (using the [CLS] token followed by a dropout layer), but replaced the default head with a small feedforward neural network consisting of the following layers:

- Linear: `hidden_size` $\rightarrow$ 128
- ReLU activation function
- Dropout (p = 0.3)
- Linear: 128 $\rightarrow$ 3

By selecting the checkpoint with the best validation accuracy during training, RoBERTa with the custom head achieved the best results in our experiments, reaching an **F1-score of 0.7982** and an **accuracy of 0.8100** on the validation set (Tab. 3) (Fig. 3).

3 Experiments

3.1 Non-LM

At the beginning, we used a more straightforward method by creating three reference vectors, one for each class. We also added the normalized numerical features at the end of the embedding vector. Each new item was classified based on which reference vector it was most similar to, using cosine similarity.
We first tried to train a *Word2Vec* model with Gensim, but the dataset was too small, and the results were not good. So, we decided to use the pretrained SpaCy model *en_core_web_lg*, which gave better results since it's trained on a much larger corpus (Tab. 5).
However, this method still performed poorly, especially for the *exclusive* class, which had lower precision and recall. These results showed the need for a different strategy, which led us to develop the two-step method described above.

3.2 LM

First, we tested several transformer models from the BERT family by doing some light hyperparameter tuning and training each of them on the same dataset. The best results for each model are shown in Table 6. Among them, RoBERTa and DistilBERT offered the best balance between accuracy, F1-score, and training time. For this reason, we decided to focus on these two models. To improve their performance, we replaced the default classification heads with custom ones. Our goal was to improve prediction accuracy and reduce model uncertainty, as shown by a lower validation loss. We saved the checkpoints that gave the highest validation accuracy during training. In both cases, the custom heads performed better than the standard ones in terms of accuracy and F1-score (see Tab. 3 and Tab. 4). In the end, RoBERTa achieved the best results, so we selected it as our final model. Our best-performing model achieved an accuracy of 0.8233 and an F1-score of 0.8117. However, these results were obtained without properly setting the seed, making the training process non-reproducible. For this reason, we chose to use a different model, despite its slightly lower performance.

4 Results

The LM-based model (RoBERTa with a custom classification head) clearly outperforms the non-LM model. It achieves an accuracy of 81.0% and an F1 of 79.8%, compared to 76.0% for both metrics in the non-LM approach. This shows that transformer-based models are more effective for this classification task.
By analyzing the results on validation sets it's possible to notice that RoBERTa is especially better at recognizing "cultural representative" items, identifying 11 more correct cases than the non-LM model (Fig. 4). It also makes fewer mistakes in "cultural agnostic" and "cultural representative" classes overall, as shown in the error analysis charts (Fig. 5).
Although the non-LM method still gives reasonable results, it tends to overpredict the "cultural exclusive" class and struggles more with abstract or complex concepts. In some cases, such as sports teams or well-known figures, it performs slightly better than the LM model. When the two models disagree (about 21.6% of the time), the LM model usually makes the more accurate choice (Fig. 6).
Finally, it's important to note that the dataset contains some ambiguous or possibly incorrect labels, which may affect the evaluation (and also reflect the subjective nature of cultural classification).

5 Appendix

Feature Type	Extracted Information
Textual (Wikidata)	Instance_of (P31)
	Country (P17)
	Country_of_origin (P495)
	English description
Textual (Wikipedia)	Categories
Numerical (Wikipedia)	Average daily views
	Word count (English page)
	Number of Wikipedia pages
Numerical (Wikidata)	Number of Wikidata properties

Table 1: Summary of features extracted from Wikidata and Wikipedia.

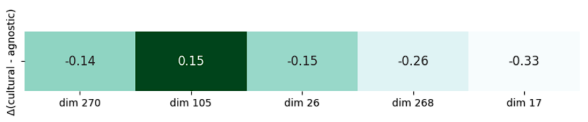

Figure 1: Top 5 dimension differences between class embeddings.

Class	Precision	Recall	F1-score	Support
Agnostic	0.82	0.90	0.86	117
Exclusive	0.65	0.72	0.68	76
Representative	0.78	0.64	0.70	107
Accuracy			0.76	300
Macro Avg	0.75	0.75	0.75	300
Weighted Avg	0.76	0.76	0.76	300

Table 2: Classification report for the final non-LM model.

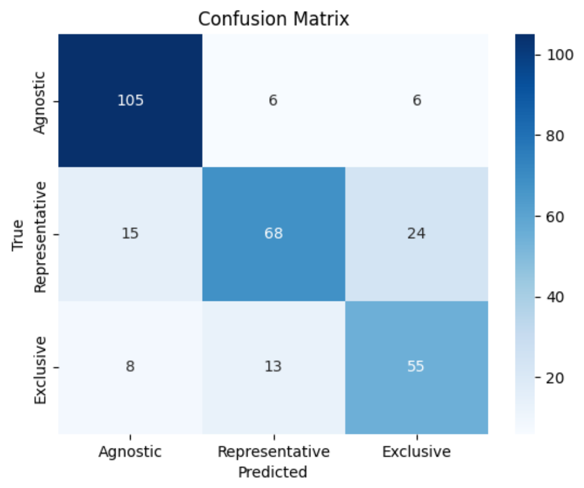

Figure 2: Confusion matrix of the final non-LM.

Epoch	Train Loss	Val Loss	Accuracy	F1
1	0.9552	0.6616	0.7433	0.7315
2	0.6110	0.6205	0.7266	0.7021
3	0.4857	0.5994	0.7500	0.7327
4	**0.4083**	**0.5385**	**0.8100**	**0.7982**
5	0.400	0.6184	0.7900	0.7779
6	0.2917	0.6626	0.8000	0.7857
7	0.2547	0.7005	0.7866	0.7734

Table 3: Training of RoBERTa with custom head. Batch size=16, warmup steps=500, weight decay=0.01, learning rate=1e-5. Early stopping with patience=3.

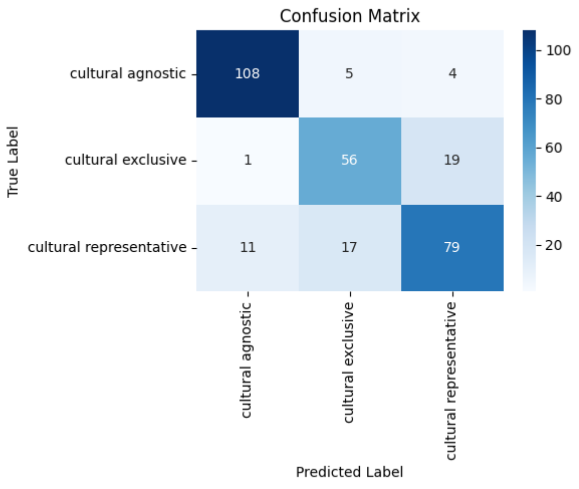

Figure 3: Confusion matrix of the LM-model.

Epoch	Train Loss	Val Loss	Accuracy	F1
1	0.993300	0.573806	0.773333	0.751730
2	**0.531000**	**0.681223**	**0.743333**	**0.723235**
3	0.427300	0.691100	0.740000	0.724633
4	0.349800	0.888932	0.736667	0.718184

Table 4: Training of DistilBERT with custom head. Batch size=8, warmup steps=500, weight decay=0.01, learning rate=1e-5, dropout=0.4. Early stopping with patience=3.

Class	Word2Vec-gensim			SpaCy		
	Precision	Recall	F1	Precision	Recall	F1
Agnostic	0.83	0.78	0.80	0.84	0.83	0.83
Exclusive	0.52	0.63	0.57	0.47	0.55	0.51
Representative	0.55	0.50	0.52	0.61	0.53	0.57
Accuracy	0.64			0.65		
Macro Avg	0.63	0.63	0.63	0.64	0.64	0.64
Weighted Avg	0.65	0.64	0.64	0.66	0.65	0.66

Table 5: Comparison between SpaCy and Word2Vec in the first approach.

Model	**Train Loss**	**Val Loss**	**F1**	**Accuracy**
DeBERTa	0.3098	0.8765	0.7785	0.7900
XLNet	0.4064	0.7721	0.7800	0.7933
ALBERT	0.6545	0.6025	0.7300	0.7466
DistilBERT	0.3199	0.9273	0.7507	0.7667
RoBERTa	0.0482	1.2410	0.7966	0.7832

Table 6: Performance comparison of different models with 3 training epochs.

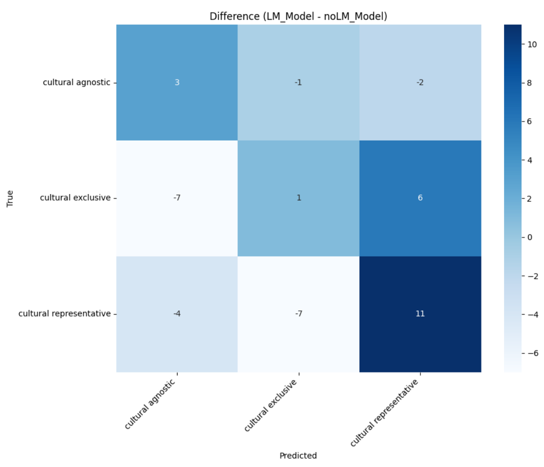

Figure 4: Confusion matrix of the difference of the predictions of the two models.

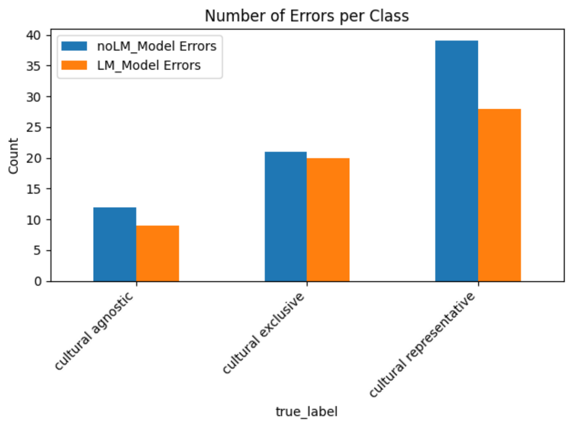

Figure 5: Bar plot of models errors in disintiguishing different classes.

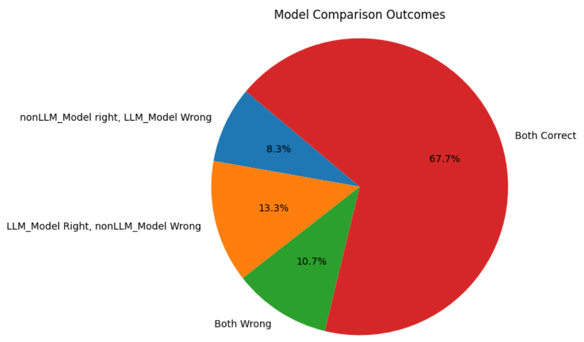

Figure 6: Bar plot of models errors in disintiguishing different classes.