Evaluating Parameter-Efficient Fine-Tuning for Character Classification in Dialogue Data: A Case Study on *The Big Bang Theory*

Iacopo Stracca
i.stracca@studenti.unipi.it

Abstract

We investigate speaker identification from short, context-free utterances in dialogue data, using transcripts from The Big Bang Theory. Treating the task as multi-class classification over SBERT sentence embeddings, we compare frozen representations with two parameter-efficient fine-tuning strategies: last-layer unfreezing and Low-Rank Adaptation (LoRA). Our experiments show that both fine-tuning methods yield modest but consistent improvements over frozen baselines. Additionally, we find that weighted loss functions help mitigate class imbalance, improving F1 scores for minority characters. Finally, a qualitative embedding analysis across seasons reveals interpretable shifts in linguistic style aligned with character development. These results underscore the potential of lightweight fine-tuning approaches in low-context, imbalanced speaker classification settings.

1 Introduction

Identifying the speaker of an utterance in a multi-character setting is a challenging problem at the intersection of natural language processing, authorship attribution, and dialogue modeling. In contrast to speaker identification in multimodal contexts—where audio and visual cues are available—text-only speaker recognition must rely exclusively on linguistic features, style, and lexical choices to make accurate predictions. This is particularly interesting in fictional dialogues, such as those in TV shows, where characters exhibit distinct personalities and idiosyncratic language patterns.

The recent availability of pre-trained sentence embedding models like Sentence-BERT (SBERT) has significantly improved our ability to represent utterances semantically in a compact vector space. However, these models are typically trained on general-purpose sentence similarity tasks and may not optimally capture the subtle stylistic features needed for fine-grained character identification. This motivates the investigation of whether fine-tuning such models—particularly through parameter-efficient strategies—can lead to significant improvements in classification accuracy.

In this project, we focus on the task of single-utterance character identification using dialogue data from the sitcom *The Big Bang Theory*. Our goal is to assess the effectiveness of different modeling strategies for improving speaker classification performance in a constrained, low-context setting. We first evaluate a simple architecture in which utterances are embedded using a frozen SBERT model and classified using a shallow neural network. We then perform a systematic hyperparameter search to identify strong baseline models across multiple classification tasks (binary and multiclass) and loss functions (standard and weighted cross-entropy). Next, we explore whether performance can be further improved by applying end-to-end fine-tuning through parameter-efficient fine-tuning (PEFT) techniques such as LoRA and partial layer unfreezing.

Finally, we analyze the best-performing model to gain qualitative insight into how character-specific representations evolve over time. In particular, we extract utterance embeddings across all seasons and apply dimensionality reduction via t-SNE to visualize the progression of each character's linguistic footprint. This offers an interpretable view of how personality and language style may shift throughout the series.

Our contributions are as follows:

- We construct and evaluate a speaker classification pipeline based on SBERT embeddings and a shallow classifier, across multiple classification settings.
- We conduct a systematic hyperparameter search over learning rate, dropout rate, and weight decay to identify strong baseline configurations.
- We investigate the effect of end-to-end fine-tuning via PEFT methods (LoRA and last-layer unfreezing) on classification performance.
- We present a qualitative embedding analysis over time using t-SNE visualizations to explore how character representations evolve across seasons.

The rest of the report is organized as follows. Section 2 reviews related work on text-based speaker and character identification. Section 3 describes our methodology, including model architectures and fine-tuning strategies. Section 4 presents the experimental setup and results. Section 5 discusses the insights gained from the experiments. Finally, Section 6 concludes with a summary and directions for future work.

Our code is publicly available in the following GitHub repository: `https://github.com/istracca/TBBT_classification`.

2 Background

Character or speaker identification from text has recently gained traction as a task combining elements of authorship attribution and dialogue modeling. While many existing approaches leverage multimodal signals—such as audio, video, or subtitle timing—or rely on dialogue-specific supervision like turn-taking, our work focuses on a stricter and less-explored setting: single-utterance, text-only speaker classification with no contextual information.

A pioneering effort in this direction is the work by Ma et al. [Ma et al., 2017], who developed a CNN-based architecture to classify speakers in transcripts from the TV series *Friends*. Their model operates at the document level, using a multi-document CNN that incorporates local context by concatenating surrounding utterances, and even experiments with merging all utterances from the same speaker within a scene. They demonstrate that including context significantly boosts performance, achieving around 34% accuracy. However, their approach is heavily dependent on intra-scene continuity and scene-level supervision, whereas we restrict our attention to individual utterances treated in isolation.

More recently, Ribeiro et al. introduced *Speaker Fuzzy Fingerprints* [Ribeiro et al., 2025], a method that combines RoBERTa-based sentence embeddings with speaker-specific prototype construction via fuzzy clustering. Their approach enhances interpretability by identifying the most salient hidden units per speaker, and attains state-of-the-art performance—reaching 70.6% accuracy on *Friends* and 67.7% on *The Big Bang Theory*. Nonetheless, their method requires aggregating multiple utterances per speaker to compute class fingerprints, both during training and inference, effectively leveraging speaker-level information unavailable in our setup. In contrast, we predict the speaker of a single, context-free utterance, without any access to prior or future turns, speaker tokens, or dialogue structure.

Nguyen et al. proposed a transformer-based architecture trained on the MediaSum dataset [Nguyen et al., 2024], which consists of transcript data from interviews and news programs. Their model achieves strong performance, but the task is closer to speaker diarization than to stylistic speaker identification in fictional dialogue, and it lacks the highly stylized, personality-driven language typical of scripted TV shows.

Closer to our setup is the preprint by Sumpter and Caut [Sumpter and Caut, 2024], who use sentence embeddings from SBERT (`all-MiniLM-L6-v2`), reduced via PCA and fed into a logistic regression classifier to distinguish between pairs of characters. Although their work remains unpublished, it provides a useful benchmark. We extend their methodology in several directions: (i) we consider multi-class classification settings, (ii) we evaluate both standard and weighted loss functions to handle class imbalance, and (iii) we explore end-to-end fine-tuning of SBERT through parameter-efficient fine-tuning (PEFT) techniques, namely *Low-Rank Adaptation (LoRA)* and partial *layer unfreezing*.

To the best of our knowledge, no prior work has systematically explored PEFT strategies in the context of character identification from dialogue using SBERT in an utterance-level, end-to-end fine-tuning framework.

3 Method

This section presents the methodological pipeline adopted in our study, which is divided into two main phases. First, we train a neural classifier on top of frozen SBERT embeddings (Section 3.5). Then, we apply parameter-efficient fine-tuning (PEFT) techniques to optimize the entire model in a controlled fashion (Section 3.6). Additionally, we perform a qualitative analysis of the embedding space by visualizing mean finetuned-SBERT embeddings per character and season using t-SNE (Section 3.7).

3.1 Dataset Preparation

We use the publicly available transcript dataset from Kaggle[1], which contains complete dialogue transcriptions from all episodes of the first 10 seasons of the TV series *The Big Bang Theory*. Each utterance is annotated with the speaking character, along with metadata such as season and episode number.

For our experiments, we retain only utterances spoken by the seven main characters: *Sheldon*, *Leonard*, *Penny*, *Howard*, *Raj*, *Amy*, and *Bernadette*. This results in a dataset of 44,966 utterances, albeit with an uneven distribution across characters. Figure 1 shows the number of utterances per character, highlighting the dominant role of the original core cast (*Sheldon*, *Leonard*, and *Penny*). Figure 2 further illustrates the evolution of character presence across seasons, capturing the gradual integration of *Amy* and *Bernadette* into the main storyline.

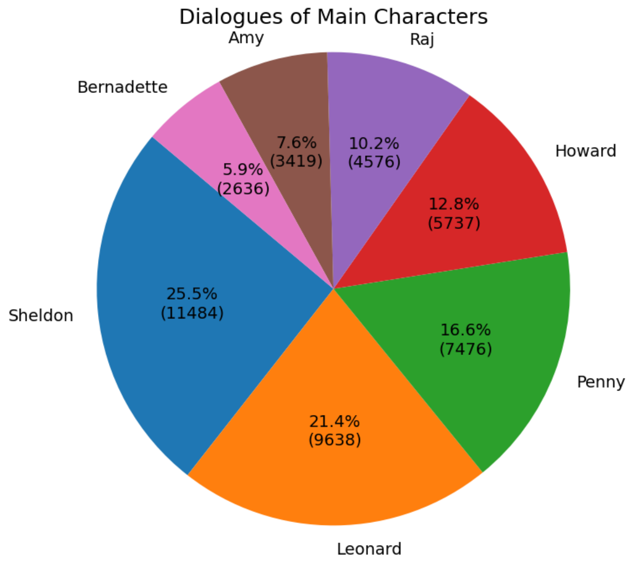


Figure 1: Distribution of utterances among the seven main characters.

[1]`https://www.kaggle.com/code/lydia70/big-bang-theory-tv-show`

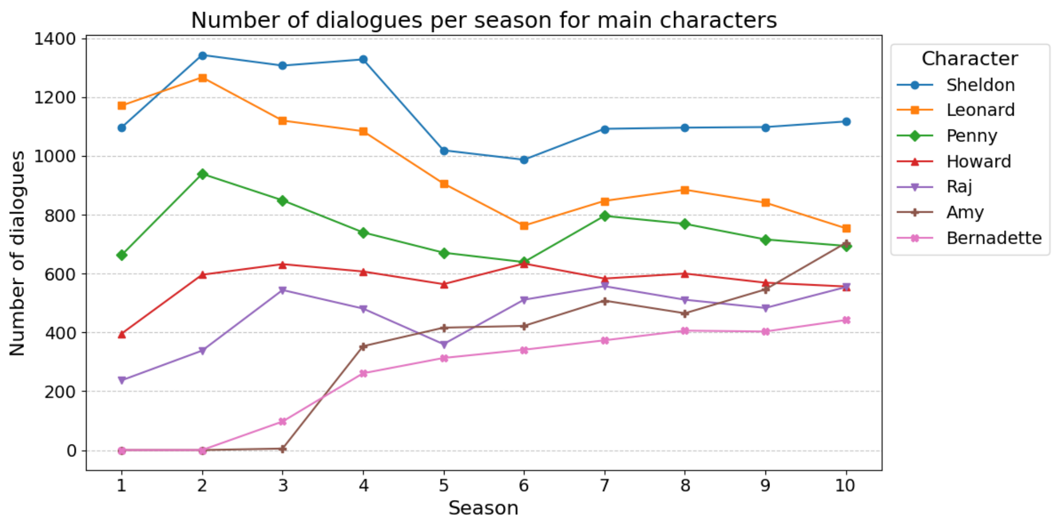


Figure 2: Character-wise evolution of utterance counts across seasons.

3.2 Sentence Embeddings

Each utterance is encoded using the `all-MiniLM-L6-v2`[2] model from the Sentence-BERT (SBERT) family [Reimers and Gurevych, 2019], which maps sentences to fixed-size 384-dimensional embeddings. We do not apply any further transformation or normalization to the embeddings, as they are already optimized for downstream tasks and empirically well-behaved in terms of geometry.

3.3 Task Formulation

We evaluate model performance across three speaker classification tasks of varying difficulty:

- **Multi-class classification**: Identify the speaker among the seven main characters. This represents the most challenging scenario due to significant class imbalance and overlapping stylistic traits across speakers.
- **Binary classification (Sheldon vs. Penny)**: A simpler task that contrasts two characters who are linguistically and behaviorally quite distinct.
- **Binary classification (Sheldon vs. Leonard)**: A somewhat more difficult binary task, involving two characters with more similar linguistic styles and social roles.

These tasks allow us to probe model robustness both in coarse-grained and fine-grained discrimination scenarios.

3.4 Loss Functions

To evaluate the effect of class imbalance, we experiment with two versions of the cross-entropy loss: the standard (uniformly weighted) loss, and a weighted variant in which each class is weighted inversely to its frequency in the training set. This allows us to assess how sensitive the classifier is to imbalanced data distributions.

Let N be the number of training samples, $\hat{\mathbf{p}}^{(i)} \in \mathbb{R}^C$ the predicted probability vector for sample i, and $y^{(i)} \in \{1, \ldots, C\}$ its ground-truth label.

The **standard cross-entropy loss** over the dataset is defined as:

$$\mathcal{L}_{\text{standard}} = -\frac{1}{N} \sum_{i=1}^{N} \log \hat{p}_{y^{(i)}}^{(i)} \tag{1}$$

[2]`https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`

In the **weighted version**, each class c is assigned a weight w_c inversely proportional to its relative frequency f_c. The loss becomes:

$$\mathcal{L}_{\text{weighted}} = -\frac{1}{N} \sum_{i=1}^{N} w_{y^{(i)}} \cdot \log \hat{p}^{(i)}_{y^{(i)}} \tag{2}$$

where the class weights w_c are given by:

$$w_c = \frac{1}{f_c} \tag{3}$$

This weighting scheme penalizes errors on minority classes more heavily, encouraging the model to better account for underrepresented categories.

We use the implementation provided by PyTorch's `CrossEntropyLoss` module, which supports weighted loss via the `weight=` argument.

3.5 Baseline Classifier Architecture

We first train a feedforward neural network on top of frozen SBERT embeddings. The architecture consists of a single hidden layer with 128 units and ReLU activation, followed by dropout and an output layer. The output size matches the number of classes in the task. We perform a grid search over learning rate, dropout rate, and weight decay for each task/loss combination, yielding a robust baseline.

3.6 Parameter-Efficient Fine-Tuning

In this phase, we perform end-to-end training of the entire model, composed of the SBERT encoder combined with the same classification head described in Section 3.5 (a feedforward neural network with one hidden layer of 128 units, ReLU activation, and dropout).

We experiment with two parameter-efficient fine-tuning strategies:

- **LoRA (Low-Rank Adaptation)** [Hu et al., 2021]: trainable low-rank adapters are inserted within the attention layers of the SBERT encoder, enabling efficient adaptation with a minimal number of additional parameters.
- **Last-Layer Fine-Tuning**: all layers of SBERT are frozen except for the last Transformer layer, which is jointly trained with the classification head.

Both methods aim to improve model performance while reducing the computational cost and potential instability associated with full fine-tuning. We apply these techniques systematically across all task and loss function configurations to assess their effectiveness.

3.7 Character-wise Embedding Visualization

To qualitatively assess how well the finetuned-SBERT embeddings capture speaker-specific linguistic traits, we compute the mean embedding of each character's utterances for every season. This results in a set of aggregated embeddings that summarize each character's language use over time. We then apply t-distributed Stochastic Neighbor Embedding (t-SNE) to project these 384-dimensional vectors into a 2D space for visualization. This procedure allows us to inspect the temporal evolution of character representations and detect stylistic shifts across seasons.

4 Experimental Analysis

4.1 Tasks

As described in Section 3.3, we evaluate speaker classification across three distinct tasks: (i) a 7-way multi-class classification over the main characters of *The Big Bang Theory*, and two binary classification tasks: (ii) Sheldon vs. Penny, and (iii) Sheldon vs. Leonard. These setups allow us to examine model behavior under varying levels of complexity and class similarity.

As a sort of *bonus task*, we also perform a qualitative analysis of the embedding space as described in Section 3.7 by computing the mean finetuned-SBERT embeddings per character and season, and visualizing them using t-SNE. This allows us to explore whether the model captures shifts in characters' linguistic style over time.

4.2 Experimental Settings

Frozen SBERT. In the first phase, we train a neural classifier on top of fixed SBERT embeddings. We perform a grid search over the following hyperparameters: learning rate $\in \{1\text{e-}5, 5\text{e-}5, 1\text{e-}4, 5\text{e-}4\}$, dropout rate $\in \{0.0, 0.1, 0.2\}$, and weight decay $\in \{0.0, 1\text{e-}4, 5\text{e-}4, 1\text{e-}3\}$.

LoRA Fine-Tuning. For parameter-efficient fine-tuning using LoRA, we conduct a grid search over the rank parameter $r \in \{8, 16, 32\}$. Following common practice—and due to limited computational resources—we fix the LoRA scaling factor to $\alpha = 2r$. Weight decay is varied within $\{0.0, 1\text{e-}4\}$, while the learning rate is kept constant at 1e-4 and the dropout rate fixed at 0.1 for both the classifier and LoRA layers.

Last-Layer Tuning. In this setting, all layers of the SBERT encoder are frozen except the last Transformer layer, which is fine-tuned jointly with the classification head. We fix the learning rate at 1e-4 and the dropout rate at 0.1, performing a grid search solely over the weight decay parameter $\in \{0.0, 1\text{e-}4\}$.

Common Training Details. All models are trained using the Adam optimizer with a batch size of 32. Early stopping is applied based on validation loss, with patience set to 10 epochs and a maximum of 1000 epochs for the frozen SBERT experiments, and patience set to 3 epochs with a maximum of 20 epochs for the LoRA and last-layer fine-tuning experiments. Data splits are stratified and divided into training (80%), validation (10%), and test (10%) sets to preserve class distributions.

Character Evolution through Seasons. We select one of the best-performing fine-tuned SBERT models and use it to recompute the embeddings of all utterances across the entire dataset (training, validation, and test combined). For each main character and each season, we then compute the mean embedding vector, and visualize their evolution over time using t-SNE for dimensionality reduction.

4.3 Results

Complete results of the grid search experiments are provided in Appendix A. Additionally, confusion matrices corresponding to the best-performing models for each combination of task and loss function are presented in Appendix B.

4.3.1 7-way classification

Table 1: Best results on the 7-way classification task for all methods.

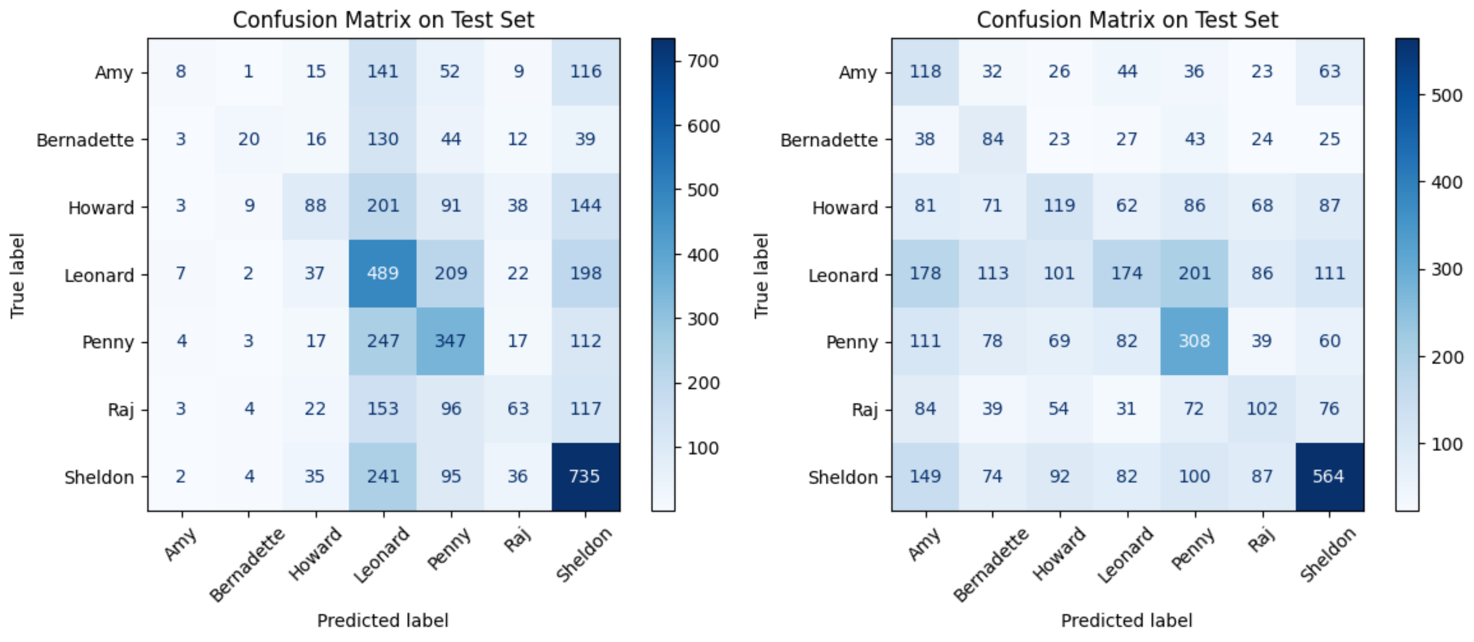
Method	Loss	Accuracy	Macro-F1	Macro-Precision	Macro-Recall	Config
Frozen SBERT	Standard CE	**0.37**	0.25	**0.39**	0.26	5e-4 / 0.1 / 5e-4
	Weighted CE	0.31	**0.27**	0.28	**0.29**	5e-4 / 0.1 / 5e-4
LoRA	Standard CE	**0.39**	0.28	**0.39**	0.28	32 / 64 / 1e-4
	Weighted CE	0.33	**0.29**	0.31	**0.31**	16 / 32 / 0
Last-layer	Standard CE	**0.39**	0.28	**0.37**	0.29	1e-4
	Weighted CE	0.33	**0.29**	0.30	**0.31**	1e-4

Table 1 summarizes the best results for each method on the 7-way classification task. Both LoRA and last-layer fine-tuning outperform the frozen SBERT baseline across all metrics, confirming the advantage of updating at least part of the transformer during training. Interestingly, LoRA and last-layer fine-tuning reach the same peak accuracy (0.39) and macro-F1 (0.28). While the overall performance metrics may appear modest in absolute terms, it is important to contextualize these results. In a comparable setting, Ribeiro et al. [2025] evaluated a 7-way speaker classification task involving the same characters from *The Big Bang Theory*, using a sentence-level setup (i.e., zero context window), and reported an accuracy of 0.32 and a macro-F1 of 0.24. Although the dataset is not identical to ours, the similarity in task formulation makes the comparison meaningful.

Table 2: Per-class performance for Last-Layer + MLP on 7-way classification task

	Standard CE			Weighted CE		
Character	Prec.	Rec.	F1	Prec.	Rec.	F1
Amy	0.27	0.02	0.04	0.16	0.35	0.21
Bernadette	0.47	0.08	0.13	0.17	0.32	0.22
Howard	0.38	0.15	0.22	0.25	0.21	0.22
Leonard	0.31	0.51	0.38	0.35	0.18	0.24
Penny	0.37	0.46	0.41	0.36	0.41	0.39
Raj	0.32	0.14	0.19	0.24	0.22	0.23
Sheldon	0.50	0.64	0.56	0.57	0.49	0.53

Table 2 provides a breakdown of the classification metrics for each character under both standard and weighted cross-entropy, in the Last-Layer + MLP setting. The weighted loss substantially boosts the recall and F1 scores for under-represented characters such as *Amy* (recall from 0.02 to 0.35), *Bernadette* (0.08 $\rightarrow$ 0.32), and *Raj* (0.14 $\rightarrow$ 0.22), showing a clear shift toward balanced predictions.



(a) Standard Cross-Entropy

(b) Weighted Cross-Entropy

Figure 3: Confusion matrices for Last-Layer + MLP on the 7-way classification task.

However, this improvement comes at a cost: dominant classes like *Sheldon* and *Leonard* see reduced recall (from 0.64 to 0.49 for *Sheldon*, 0.51 to 0.18 for *Leonard*), as well as drops in precision and F1. The model learns to distribute its predictions more evenly, but this rebalancing slightly compromises performance on previously dominant categories. This trade-off is clearly visible in the confusion matrices shown in Figure 3, where the weighted variant corrects many false negatives for minority characters at the expense of introducing more confusion among the majority ones.

Overall, the weighted loss is effective in promoting inclusiveness across classes, especially when macro-level metrics like F1 and recall are prioritized over raw accuracy. Nevertheless, the observed drop in performance for high-frequency classes, especially for *Leonard*, suggests that the weighting scheme may be acting too aggressively. Alternative approaches—such as *focal loss* Lin et al. [2017] or more refined *reweighting heuristics* Cui et al. [2019]—could offer a more balanced trade-off by penalizing majority classes less severely while still addressing class imbalance. Another potential strategy is *undersampling* the most frequent classes Chawla et al. [2004], however, this would further reduce the size of an already limited dataset, potentially impairing the model's generalization ability.

4.3.2 Binary classification tasks

Table 3: Best results on the binary classification task (Sheldon vs. Penny)

Method	Loss	Accuracy	Macro-F1	Macro-Precision	Macro-Recall	Config
Frozen SBERT	Standard CE	**0.75**	0.73	**0.74**	0.73	5e-4 / 0.2 / 5e-4
	Weighted CE	0.73	0.73	0.73	0.73	1e-4 / 0 / 5e-4
LoRA	Standard CE	0.76	0.74	**0.76**	0.74	32 / 64 / 1e-4
	Weighted CE	0.76	**0.76**	0.75	**0.77**	16 / 32 / 0
Last-layer	Standard CE	**0.76**	**0.75**	**0.76**	0.74	0
	Weighted CE	0.74	0.73	0.73	**0.75**	1e-4

Table 4: Best results on the binary classification task (Sheldon vs. Leonard)

Method	Loss	Accuracy	Macro-F1	Macro-Precision	Macro-Recall	Config
Frozen SBERT	Standard CE	**0.70**	**0.70**	**0.70**	**0.70**	5e-4 / 0.1 / 1e-3
	Weighted CE	0.69	0.69	0.69	0.69	1e-4 / 0.2 / 1e-3
LoRA	Standard CE	0.72	0.72	0.72	0.72	8 / 16 / 1e-4
	Weighted CE	0.72	0.72	0.72	0.72	16 / 32 / 0
Last-layer	Standard CE	**0.71**	**0.71**	0.71	0.71	0
	Weighted CE	0.70	0.70	0.71	0.71	1e-4

Tables 3 and 4 show the results of the two binary classification tasks. Unlike what was seen for the 7-way task, here the difference between the two losses is much less evident, especially for the *Sheldon vs. Leonard* task. This is because the two classes in each of these tasks are just slightly unbalanced, different from what happened in the 7-way task. The best results achieved are an accuracy of 0.76 and a macro-F1 of 0.76 for *Sheldon vs Penny* and an accuracy of 0.72 and a macro-F1 of 0.72 for *Sheldon vs Leonard* (in both cases with LoRA). Comparing to our results, Sumpter and Caut [2024] obtained an accuracy of 0.73 for *Sheldon vs Penny* and an accuracy of 0.68 for *Sheldon vs Leonard*.

4.3.3 Character Evolution through Seasons

Figure 4 shows the mean embeddings of the entire dataset, grouped by character and season, computed using the Last-Layer finetuned SBERT described in Section 4.3.1. We selected the model trained with the weighted cross-entropy loss, as it is expected to provide a more balanced representation across all characters, rather than favoring those with the most dialogue.

From this overview, we observe that the trajectories of all characters remain well separated over time, evolving without intersecting each other. Notably, *Raj*, *Howard*, and *Leonard*—the group of nerdy scientists—are located relatively close to one another. Sheldon, a very distinctive character, is positioned far from all others and appears to be one of the least changed characters across the seasons.

If we want to indulge in some narrative interpretation, we can observe that Leonard, Penny's partner, is the closest character to her; similarly, Bernadette, Howard's partner, is the closest female character to him; and Amy, Sheldon's partner, is the character nearest to him.

We also plot in Figure 5 the evolution of some characters separately to highlight the directions of their trajectories. For instance, Amy starts near Sheldon's region and gradually moves towards the cluster of more *typical* characters. Likewise, Howard and Raj begin quite far from the female characters (*Penny* and *Bernadette*) and progressively approach them, reflecting the storyline where, at the beginning of the series, they struggle to interact with women.

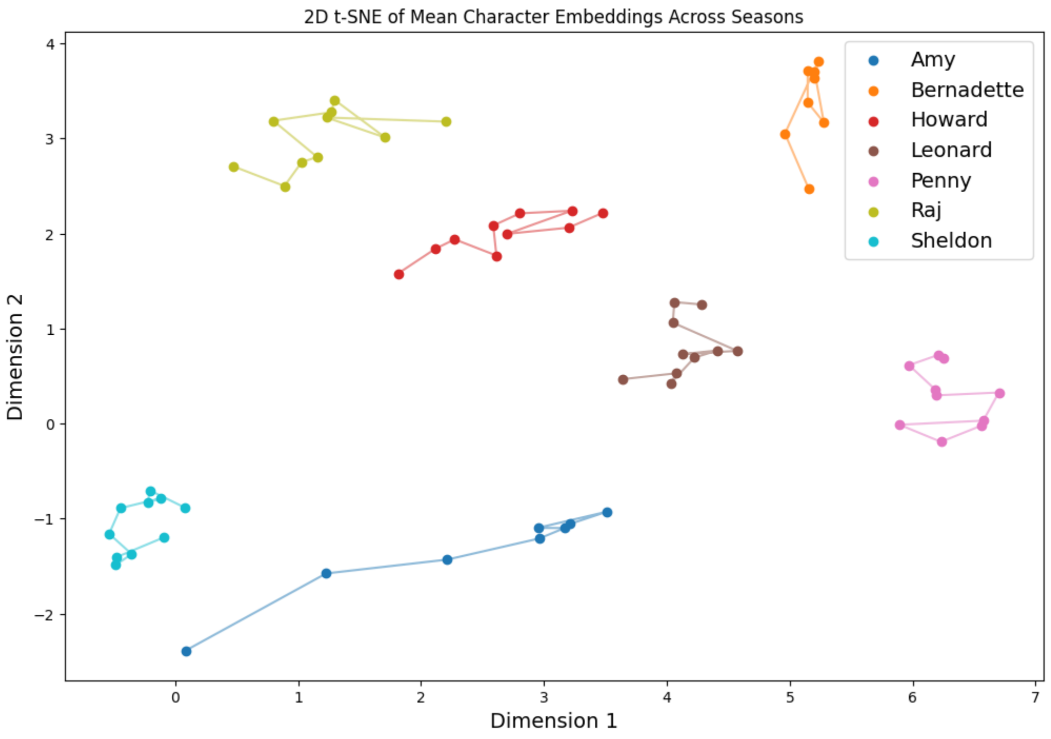
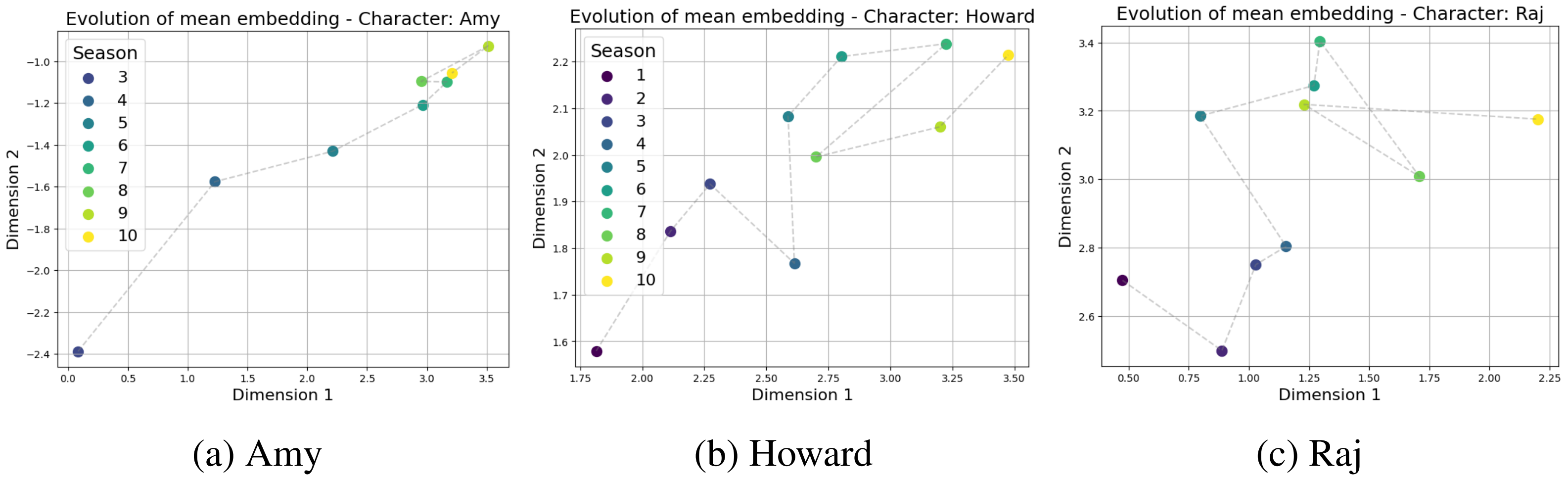


Figure 4: Character evolution over seasons visualized through 2D t-SNE of mean finetuned-SBERT embeddings.

We also report, in Appendix C, the per-season cosine distances between the mean phrase embeddings of all character pairs.



(a) Amy (b) Howard (c) Raj

Figure 5: t-SNE visualizations of mean SBERT embeddings by character across the seasons.

5 Discussion

When I began this project, I had higher expectations for the potential outcomes, and I initially felt somewhat disappointed by the final results. However, several factors may explain these limitations:

- The classification head was a very simple MLP, consisting of just a single hidden layer with 128 neurons. Although a grid search was conducted during training, it was limited due to time constraints and did not explore the full space of possible hyperparameters. For instance, techniques such as learning rate scheduling, momentum, or more advanced regularization methods were not considered.
- The fine-tuning procedure was designed to be extremely parameter-efficient, in order to run on hardware with limited GPU capabilities. This imposed strict constraints on the hyperparameter search space, both in terms of the number of parameters and the range of values explored.
- The task turned out to be more difficult than expected. Many utterances in the series are short and generic, making it hard to recognize the speaker—even for a devoted fan—without additional context. Sumpter and Caut [2024] attempted to assess the intrinsic difficulty of our two binary classification tasks by asking two human experts to classify a sample of 100 utterances. The reported accuracies were 0.71 and 0.74 for *Sheldon vs. Penny*, and 0.76 and 0.72 for *Sheldon vs. Leonard*, suggesting that even humans struggle with this task.
- The limited size of the dataset further complicates the task. Even if we were able to remove all the neutral utterances—common to all characters and essentially noise for the classifier—we would end up with an even smaller dataset. This scarcity of data makes it challenging to train more complex models without overfitting. From a theoretical perspective, the Vapnik–Chervonenkis (VC) dimension framework suggests that models with higher capacity require more data to generalize effectively Vapnik [1998].

Beyond the aforementioned limitations, we observed a consistent—albeit modest—performance improvement in models that included fine-tuning (LoRA, last-layer) compared to those relying on a frozen SBERT. This suggests that, despite being highly parameter-efficient, fine-tuning can still provide tangible benefits when addressing a highly domain-specific task.

6 Conclusions

In this work, we investigated different fine-tuning strategies for speaker identification based on the dialogue lines of *The Big Bang Theory* characters, comparing frozen SBERT embeddings, LoRA, and last-layer fine-tuning approaches. Our experiments on a challenging 7-way classification task showed that both LoRA and last-layer fine-tuning consistently outperform the frozen baseline, confirming the benefit of adapting transformer parameters even partially.

While the overall performance metrics remain moderate, this is consistent with the complexity of the task and the sentence-level setup, which relies on very limited context. Notably, our best models improved over prior comparable work [Ribeiro et al., 2025] by a meaningful margin.

The use of weighted cross-entropy effectively alleviated class imbalance issues, significantly boosting recall and F1 scores for minority classes at the cost of some degradation in majority class performance. This trade-off highlights the challenge of balancing inclusiveness and accuracy in imbalanced datasets, and suggests avenues for future work exploring alternative loss functions and reweighting schemes.

Our qualitative analysis of character embeddings over seasons reveals distinct and evolving trajectories aligned with the narrative arcs of the series, demonstrating the potential of embedding-based representations to capture character-specific linguistic style changes over time.

Future work may explore more sophisticated fine-tuning methods and classification architectures to further enhance speaker identification performance. Moreover, investigating alternative imbalance handling techniques could yield improved trade-offs between precision and recall across all classes.

Overall, this study lays a foundation for leveraging transformer-based embeddings in speaker identification within TV series dialogues, providing insights and baselines for subsequent research in this domain.

References

Nitesh V. Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*, 6(1):1–6, June 2004. ISSN 1931-0145. doi:

10.1145/1007730.1007733. URL `https://doi.org/10.1145/1007730.1007733`.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples, 2019. URL `https://arxiv.org/abs/1901.05555`.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL `https://arxiv.org/abs/2106.09685`.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017. doi: 10.1109/ICCV.2017.324.

Kaixin Ma, Catherine Xiao, and Jinho Choi. Text-based speaker identification on multiparty dialogues using multi-document convolutional neural networks. pages 49–55, 07 2017. doi: 10.18653/v1/P17-3009.

Minh Nguyen, Franck Dernoncourt, Seunghyun Yoon, Hanieh Deilamsalehy, Hao Tan, Ryan Rossi, Quan Hung Tran, Trung Bui, and Thien Huu Nguyen. Identifying speakers in dialogue transcripts: A text-based approach using pretrained language models, 2024. URL `https://arxiv.org/abs/2407.12094`.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. pages 3973–3983, 01 2019. doi: 10.18653/v1/D19-1410.

Rui Ribeiro, Luisa Coheur, and Joao Carvalho. Speaker fuzzy fingerprints: Benchmarking text-based identification in multiparty dialogues, 04 2025.

David Sumpter and Samuel Caut. A test of stochastic parroting in a generalisation task: Predicting the characters in tv series, 2024. URL `https://openreview.net/forum?id=XkMCKoHNCD`. Preprint. Submitted and rejected. Available on OpenReview.

Vladimir N. Vapnik. *Statistical learning theory*. Wiley, 1998.

A Full grid search results

A.1 Frozen SBERT

Table 5: Frozen SBERT, Cross Entropy, 7-way classifier

learning_rate	dropout_rate	weight_decay	epoch	train_loss	val_loss	train_acc	val_acc
5e-04	0.1000	5e-04	98	1.5351	1.6786	0.4291	0.3492
1e-04	0.2000	5e-04	245	1.5756	1.6860	0.4108	0.3498
1e-04	0.1000	5e-04	231	1.5708	1.6869	0.4123	0.3503
5e-04	0.2000	5e-04	62	1.5860	1.6871	0.4052	0.3522
5e-05	0.1000	5e-04	452	1.5720	1.6878	0.4119	0.3449
1e-04	0.2000	1e-04	113	1.5437	1.6884	0.4204	0.3416
5e-04	0.0000	5e-04	53	1.5642	1.6907	0.4162	0.3414
5e-04	0.2000	1e-04	31	1.5134	1.6913	0.4311	0.3487
1e-04	0.1000	1e-04	110	1.5350	1.6918	0.4242	0.3467
1e-04	0.0000	5e-04	182	1.5848	1.6918	0.4061	0.3478
5e-05	0.2000	5e-04	249	1.6178	1.6924	0.3883	0.3421
5e-05	0.0000	5e-04	344	1.5953	1.6924	0.4018	0.3434
5e-05	0.2000	1e-04	196	1.5646	1.6928	0.4109	0.3489
1e-05	0.1000	1e-04	883	1.5730	1.6930	0.4085	0.3491
5e-05	0.2000	0e+00	159	1.5695	1.6930	0.4078	0.3496
1e-04	0.2000	0e+00	86	1.5618	1.6933	0.4102	0.3492
1e-05	0.2000	1e-04	899	1.5793	1.6935	0.4042	0.3434
1e-04	0.1000	0e+00	81	1.5611	1.6938	0.4119	0.3438
5e-05	0.1000	1e-04	233	1.5310	1.6946	0.4263	0.3418
5e-05	0.0000	1e-04	192	1.5571	1.6948	0.4157	0.3423
1e-05	0.0000	1e-04	1000	1.5610	1.6948	0.4114	0.3461
1e-05	0.2000	0e+00	605	1.6045	1.6950	0.3907	0.3423
1e-04	0.0000	1e-04	104	1.5388	1.6951	0.4245	0.3441
1e-05	0.2000	5e-04	1000	1.6296	1.6955	0.3835	0.3401
5e-05	0.0000	0e+00	161	1.5639	1.6960	0.4112	0.3489
1e-05	0.0000	0e+00	773	1.5699	1.6961	0.4094	0.3465
5e-04	0.1000	1e-04	25	1.5332	1.6963	0.4227	0.3440
1e-05	0.1000	5e-04	999	1.6295	1.6963	0.3794	0.3372
5e-05	0.1000	0e+00	155	1.5689	1.6973	0.4081	0.3423
5e-04	0.0000	0e+00	17	1.5416	1.6973	0.4210	0.3474
5e-04	0.0000	1e-04	22	1.5461	1.6984	0.4214	0.3429
1e-05	0.1000	0e+00	586	1.6009	1.6991	0.3927	0.3412
1e-04	0.0000	0e+00	88	1.5483	1.6991	0.4166	0.3474
1e-05	0.0000	5e-04	997	1.6351	1.7001	0.3773	0.3361
5e-04	0.0000	1.0e-03	88	1.6583	1.7006	0.3703	0.3408
5e-04	0.1000	1.0e-03	72	1.6655	1.7022	0.3668	0.3414
1e-04	0.0000	1.0e-03	247	1.6557	1.7024	0.3713	0.3374
5e-04	0.1000	0e+00	20	1.5218	1.7024	0.4318	0.3382
5e-04	0.2000	1.0e-03	63	1.6742	1.7036	0.3613	0.3356
5e-05	0.2000	1.0e-03	329	1.6619	1.7037	0.3690	0.3400
5e-04	0.2000	0e+00	26	1.4984	1.7037	0.4385	0.3459
1e-04	0.1000	1.0e-03	134	1.6632	1.7040	0.3669	0.3380
1e-04	0.2000	1.0e-03	117	1.6689	1.7057	0.3626	0.3358
5e-05	0.0000	1.0e-03	293	1.6662	1.7065	0.3633	0.3338
5e-05	0.1000	1.0e-03	188	1.6695	1.7071	0.3606	0.3338
1e-05	0.1000	1.0e-03	1000	1.6682	1.7071	0.3634	0.3351
1e-05	0.2000	1.0e-03	1000	1.6699	1.7076	0.3633	0.3345
1e-05	0.0000	1.0e-03	1000	1.6694	1.7086	0.3623	0.3323

Table 6: Frozen SBERT, Weighted Cross Entropy, 7-way classifier

learning_rate	dropout_rate	weight_decay	epoch	train_loss	val_loss	train_acc	val_acc
5e-04	0.2000	5e-04	47	1.6729	1.8102	0.3555	0.2972
5e-04	0.1000	5e-04	47	1.6487	1.8136	0.3640	0.2906
5e-04	0.2000	1e-04	20	1.6698	1.8156	0.3561	0.2829
5e-04	0.0000	5e-04	38	1.6779	1.8172	0.3561	0.2829
1e-04	0.2000	5e-04	112	1.7217	1.8174	0.3344	0.2877
5e-05	0.2000	5e-04	211	1.7231	1.8175	0.3362	0.2860
1e-04	0.1000	5e-04	164	1.6866	1.8179	0.3513	0.2875
5e-04	0.1000	1.0e-03	80	1.7673	1.8186	0.3152	0.2918
1e-04	0.0000	5e-04	123	1.7144	1.8190	0.3397	0.2871
5e-05	0.1000	5e-04	264	1.7086	1.8194	0.3420	0.2858
1e-04	0.2000	1e-04	55	1.7299	1.8197	0.3311	0.2807
5e-04	0.2000	1.0e-03	54	1.7777	1.8203	0.3134	0.2829
1e-04	0.2000	0e+00	49	1.7284	1.8205	0.3304	0.2842
5e-05	0.2000	0e+00	115	1.7105	1.8210	0.3375	0.2895
5e-05	0.2000	1e-04	134	1.7090	1.8212	0.3410	0.2833
1e-05	0.2000	0e+00	462	1.7363	1.8213	0.3258	0.2838
5e-04	0.0000	1.0e-03	63	1.7680	1.8215	0.3155	0.2820
5e-05	0.0000	5e-04	249	1.7174	1.8215	0.3392	0.2886
5e-05	0.1000	0e+00	133	1.6963	1.8218	0.3444	0.2873
1e-04	0.1000	1e-04	65	1.7035	1.8221	0.3433	0.2807
5e-05	0.1000	1e-04	139	1.7023	1.8222	0.3406	0.2862
1e-05	0.2000	5e-04	736	1.7499	1.8224	0.3217	0.2818
5e-05	0.0000	1e-04	140	1.7067	1.8228	0.3411	0.2856
5e-04	0.1000	1e-04	24	1.6133	1.8229	0.3788	0.2951
1e-04	0.1000	0e+00	56	1.7079	1.8229	0.3378	0.2864
5e-04	0.0000	1e-04	18	1.6743	1.8230	0.3541	0.2860
1e-04	0.2000	1.0e-03	157	1.7682	1.8231	0.3166	0.2862
5e-05	0.2000	1.0e-03	267	1.7691	1.8233	0.3159	0.2833
1e-04	0.0000	1e-04	83	1.6770	1.8234	0.3548	0.2820
1e-04	0.1000	1.0e-03	150	1.7710	1.8238	0.3155	0.2851
1e-05	0.2000	1e-04	364	1.7595	1.8244	0.3209	0.2854
5e-04	0.1000	0e+00	17	1.6468	1.8254	0.3656	0.2922
1e-05	0.1000	1e-04	412	1.7499	1.8259	0.3225	0.2844
1e-05	0.1000	5e-04	526	1.7635	1.8261	0.3189	0.2831
1e-04	0.0000	0e+00	62	1.7075	1.8261	0.3409	0.2831
5e-04	0.2000	0e+00	17	1.6653	1.8263	0.3549	0.2840
1e-05	0.1000	0e+00	342	1.7522	1.8270	0.3206	0.2814
5e-05	0.1000	1.0e-03	147	1.7823	1.8278	0.3111	0.2822
5e-04	0.0000	0e+00	16	1.6529	1.8284	0.3621	0.2902
1e-05	0.0000	0e+00	240	1.7692	1.8284	0.3162	0.2853
1e-05	0.0000	5e-04	304	1.7785	1.8286	0.3137	0.2831
5e-05	0.0000	1.0e-03	140	1.7835	1.8286	0.3104	0.2811
1e-05	0.2000	1.0e-03	535	1.7873	1.8288	0.3092	0.2818
1e-05	0.0000	1e-04	241	1.7730	1.8291	0.3132	0.2833
1e-05	0.1000	1.0e-03	543	1.7854	1.8293	0.3114	0.2851
1e-04	0.0000	1.0e-03	68	1.7873	1.8293	0.3087	0.2813
5e-05	0.0000	0e+00	54	1.7669	1.8298	0.3172	0.2840
1e-05	0.0000	1.0e-03	461	1.7866	1.8299	0.3111	0.2838

Table 7: Frozen SBERT, Cross Entropy, Sheldon vs. Penny

learning_rate	dropout_rate	weight_decay	epoch	train_loss	val_loss	train_acc	val_acc
1e-04	0.2000	1.0e-03	247	0.4101	0.4736	0.8214	0.7604
5e-05	0.1000	1e-04	216	0.3613	0.4751	0.8504	0.7568
1e-05	0.1000	1e-04	997	0.3909	0.4752	0.8277	0.7568
1e-05	0.2000	0e+00	824	0.4053	0.4765	0.8186	0.7573
1e-04	0.2000	1e-04	107	0.3743	0.4766	0.8364	0.7609
1e-04	0.1000	5e-04	124	0.4046	0.4767	0.8200	0.7615
5e-05	0.1000	0e+00	180	0.3702	0.4769	0.8416	0.7630
1e-04	0.2000	5e-04	152	0.3844	0.4771	0.8338	0.7583
1e-04	0.1000	1e-04	92	0.3944	0.4771	0.8251	0.7516
5e-05	0.2000	1e-04	194	0.3836	0.4773	0.8330	0.7630
1e-05	0.1000	0e+00	796	0.4149	0.4774	0.8127	0.7578
5e-05	0.1000	5e-04	265	0.4039	0.4780	0.8212	0.7625
5e-05	0.2000	5e-04	254	0.4109	0.4780	0.8168	0.7589
5e-05	0.2000	0e+00	181	0.3930	0.4782	0.8231	0.7562
5e-05	0.0000	5e-04	251	0.4265	0.4786	0.8060	0.7562
1e-05	0.2000	1e-04	838	0.4174	0.4787	0.8078	0.7562
1e-04	0.1000	0e+00	93	0.3724	0.4788	0.8401	0.7562
1e-04	0.2000	0e+00	87	0.3894	0.4789	0.8306	0.7651
1e-04	0.0000	1e-04	96	0.3808	0.4790	0.8360	0.7589
1e-05	0.0000	0e+00	762	0.4072	0.4795	0.8163	0.7583
1e-05	0.2000	5e-04	999	0.4367	0.4796	0.7957	0.7557
5e-04	0.1000	1.0e-03	49	0.4231	0.4797	0.8102	0.7594
1e-05	0.1000	5e-04	999	0.4382	0.4797	0.7963	0.7557
5e-05	0.0000	0e+00	158	0.3984	0.4800	0.8229	0.7583
5e-04	0.2000	5e-04	45	0.3438	0.4801	0.8583	0.7552
1e-05	0.0000	1e-04	846	0.4003	0.4803	0.8253	0.7604
1e-04	0.1000	1.0e-03	188	0.4373	0.4804	0.8020	0.7557
1e-05	0.0000	5e-04	999	0.4376	0.4806	0.7992	0.7562
1e-04	0.0000	1.0e-03	233	0.4287	0.4808	0.8059	0.7604
5e-04	0.0000	1.0e-03	87	0.3819	0.4808	0.8403	0.7547
5e-05	0.0000	1e-04	167	0.4048	0.4808	0.8209	0.7583
1e-04	0.0000	0e+00	85	0.3848	0.4810	0.8311	0.7552
1e-04	0.0000	5e-04	156	0.3829	0.4813	0.8353	0.7536
5e-04	0.1000	5e-04	36	0.3594	0.4815	0.8487	0.7578
5e-04	0.1000	1e-04	25	0.3468	0.4827	0.8539	0.7500
5e-04	0.2000	1.0e-03	41	0.4323	0.4832	0.8017	0.7615
5e-04	0.1000	0e+00	21	0.3541	0.4837	0.8488	0.7500
5e-04	0.2000	1e-04	27	0.3451	0.4837	0.8556	0.7568
5e-05	0.2000	1.0e-03	288	0.4554	0.4838	0.7854	0.7536
5e-04	0.0000	5e-04	35	0.3768	0.4845	0.8389	0.7573
5e-04	0.0000	1e-04	18	0.3955	0.4848	0.8250	0.7620
5e-05	0.0000	1.0e-03	343	0.4571	0.4864	0.7876	0.7526
1e-05	0.0000	1.0e-03	998	0.4735	0.4878	0.7753	0.7536
5e-04	0.2000	0e+00	22	0.3584	0.4878	0.8474	0.7589
5e-05	0.1000	1.0e-03	241	0.4732	0.4883	0.7760	0.7542
5e-04	0.0000	0e+00	20	0.3534	0.4886	0.8484	0.7536
1e-05	0.2000	1.0e-03	998	0.4797	0.4887	0.7704	0.7583
1e-05	0.1000	1.0e-03	496	0.5004	0.4948	0.7542	0.7526

Table 8: Frozen SBERT, Weighted cross Entropy, Sheldon vs. Penny

learning_rate	dropout_rate	weight_decay	epoch	train_loss	val_loss	train_acc	val_acc
1e-04	0.1000	5e-04	152	0.3906	0.4867	0.8323	0.7516
1e-04	0.2000	0e+00	95	0.3881	0.4869	0.8306	0.7432
5e-05	0.1000	5e-04	269	0.4065	0.4874	0.8242	0.7604
5e-05	0.2000	5e-04	275	0.4019	0.4875	0.8267	0.7490
1e-04	0.0000	5e-04	153	0.3874	0.4878	0.8336	0.7484
1e-05	0.1000	1e-04	890	0.4135	0.4885	0.8161	0.7526
1e-04	0.0000	0e+00	86	0.3922	0.4895	0.8312	0.7500
5e-05	0.2000	1.0e-03	404	0.4296	0.4899	0.8067	0.7438
5e-05	0.2000	0e+00	186	0.3941	0.4900	0.8278	0.7458
1e-04	0.2000	5e-04	122	0.4164	0.4901	0.8140	0.7484
1e-05	0.1000	0e+00	756	0.4265	0.4901	0.8061	0.7474
1e-04	0.1000	1e-04	95	0.3928	0.4902	0.8272	0.7448
1e-04	0.2000	1.0e-03	195	0.4372	0.4902	0.8010	0.7521
1e-04	0.0000	1.0e-03	214	0.4342	0.4903	0.8045	0.7516
5e-04	0.2000	1.0e-03	58	0.4097	0.4905	0.8209	0.7490
1e-04	0.2000	1e-04	109	0.3717	0.4908	0.8401	0.7547
5e-05	0.2000	1e-04	186	0.4003	0.4908	0.8224	0.7464
5e-05	0.1000	0e+00	190	0.3810	0.4914	0.8375	0.7427
1e-05	0.2000	1e-04	748	0.4405	0.4916	0.7947	0.7536
5e-05	0.0000	5e-04	247	0.4263	0.4918	0.8102	0.7417
5e-05	0.1000	1e-04	170	0.4093	0.4918	0.8201	0.7438
1e-05	0.2000	5e-04	998	0.4446	0.4920	0.7952	0.7536
1e-05	0.0000	1e-04	785	0.4188	0.4920	0.8119	0.7490
1e-05	0.1000	5e-04	998	0.4453	0.4923	0.7950	0.7552
5e-05	0.0000	1e-04	173	0.4081	0.4925	0.8192	0.7448
5e-04	0.1000	1e-04	22	0.3747	0.4926	0.8352	0.7547
1e-04	0.1000	1.0e-03	176	0.4443	0.4926	0.7967	0.7474
1e-05	0.0000	0e+00	736	0.4252	0.4927	0.8085	0.7526
5e-04	0.2000	5e-04	40	0.3578	0.4931	0.8501	0.7427
5e-05	0.1000	1.0e-03	380	0.4456	0.4931	0.7996	0.7495
5e-04	0.1000	5e-04	27	0.4023	0.4932	0.8207	0.7406
5e-05	0.0000	0e+00	181	0.3866	0.4933	0.8343	0.7438
1e-04	0.1000	0e+00	79	0.4111	0.4934	0.8176	0.7490
5e-04	0.0000	1.0e-03	82	0.3774	0.4940	0.8448	0.7422
5e-04	0.0000	1e-04	23	0.3661	0.4942	0.8469	0.7464
1e-05	0.0000	5e-04	1000	0.4464	0.4943	0.7927	0.7500
1e-04	0.0000	1e-04	88	0.4147	0.4944	0.8167	0.7453
5e-04	0.0000	5e-04	36	0.3662	0.4949	0.8469	0.7474
1e-05	0.2000	0e+00	667	0.4422	0.4955	0.7939	0.7448
5e-04	0.1000	1.0e-03	36	0.4554	0.4960	0.7895	0.7432
5e-04	0.1000	0e+00	22	0.3490	0.4960	0.8544	0.7417
5e-04	0.2000	1e-04	27	0.3525	0.4965	0.8508	0.7391
5e-04	0.0000	0e+00	19	0.3677	0.5002	0.8440	0.7500
5e-05	0.0000	1.0e-03	237	0.4866	0.5017	0.7623	0.7469
5e-04	0.2000	0e+00	25	0.3460	0.5018	0.8527	0.7568
1e-05	0.2000	1.0e-03	997	0.4894	0.5021	0.7602	0.7458
1e-05	0.0000	1.0e-03	924	0.4931	0.5033	0.7564	0.7510
1e-05	0.1000	1.0e-03	850	0.4960	0.5036	0.7540	0.7495

Table 9: Frozen SBERT, Cross Entropy, Sheldon vs. Leonard

learning_rate	dropout_rate	weight_decay	epoch	train_loss	val_loss	train_acc	val_acc
5e-04	0.1000	1.0e-03	53	0.4726	0.5753	0.7854	0.6960
5e-04	0.0000	1.0e-03	48	0.4688	0.5778	0.7882	0.6927
1e-04	0.2000	5e-04	116	0.4655	0.5781	0.7883	0.7003
5e-04	0.2000	1.0e-03	43	0.4873	0.5784	0.7707	0.6979
5e-04	0.2000	5e-04	25	0.4735	0.5784	0.7819	0.6951
1e-04	0.1000	5e-04	102	0.4759	0.5786	0.7800	0.6993
1e-04	0.0000	1.0e-03	155	0.4939	0.5788	0.7679	0.6937
1e-04	0.1000	1e-04	63	0.4812	0.5788	0.7721	0.6979
5e-05	0.1000	5e-04	215	0.4660	0.5792	0.7863	0.6965
1e-04	0.0000	5e-04	113	0.4598	0.5795	0.7916	0.6974
1e-04	0.2000	1.0e-03	159	0.4988	0.5795	0.7626	0.6922
5e-05	0.2000	5e-04	213	0.4784	0.5799	0.7775	0.6951
5e-05	0.1000	1.0e-03	277	0.5093	0.5800	0.7528	0.6918
5e-05	0.1000	0e+00	130	0.4685	0.5808	0.7821	0.6965
1e-04	0.2000	0e+00	63	0.4811	0.5808	0.7736	0.6993
1e-04	0.1000	1.0e-03	154	0.4964	0.5810	0.7669	0.6875
5e-05	0.0000	0e+00	122	0.4705	0.5810	0.7782	0.6937
1e-05	0.0000	0e+00	578	0.4792	0.5810	0.7739	0.6908
1e-04	0.0000	0e+00	56	0.4828	0.5811	0.7691	0.6913
1e-04	0.2000	1e-04	71	0.4794	0.5811	0.7753	0.6955
1e-04	0.0000	1e-04	70	0.4668	0.5814	0.7849	0.6937
5e-05	0.0000	1e-04	131	0.4793	0.5814	0.7741	0.7022
5e-05	0.0000	5e-04	179	0.4865	0.5816	0.7703	0.6932
1e-05	0.2000	0e+00	575	0.4978	0.5816	0.7609	0.6922
5e-05	0.1000	1e-04	138	0.4726	0.5818	0.7823	0.6903
1e-05	0.2000	1e-04	499	0.5172	0.5818	0.7421	0.6828
5e-05	0.2000	1e-04	137	0.4870	0.5820	0.7692	0.6932
5e-05	0.2000	1.0e-03	251	0.5181	0.5824	0.7437	0.6955
5e-05	0.0000	1.0e-03	269	0.5100	0.5826	0.7512	0.6866
1e-04	0.1000	0e+00	72	0.4509	0.5838	0.7954	0.6922
5e-05	0.2000	0e+00	128	0.4854	0.5839	0.7671	0.6937
1e-05	0.0000	1e-04	443	0.5207	0.5841	0.7385	0.6875
5e-04	0.1000	5e-04	28	0.4435	0.5843	0.8045	0.6918
5e-04	0.0000	5e-04	29	0.4328	0.5852	0.8122	0.6970
1e-05	0.1000	0e+00	358	0.5311	0.5855	0.7316	0.6823
1e-05	0.2000	5e-04	585	0.5300	0.5857	0.7334	0.6837
1e-05	0.1000	1e-04	354	0.5389	0.5862	0.7237	0.6832
5e-04	0.0000	1e-04	17	0.4472	0.5883	0.7987	0.6932
1e-05	0.1000	5e-04	384	0.5494	0.5885	0.7160	0.6832
1e-05	0.0000	5e-04	308	0.5570	0.5911	0.7093	0.6832
5e-04	0.2000	0e+00	17	0.4532	0.5912	0.7919	0.6974
5e-04	0.1000	0e+00	18	0.4244	0.5930	0.8142	0.6880
1e-05	0.2000	1.0e-03	323	0.5673	0.5931	0.7039	0.6813
5e-04	0.2000	1e-04	21	0.4366	0.5937	0.8043	0.6880
1e-05	0.0000	1.0e-03	336	0.5658	0.5944	0.7040	0.6757
1e-05	0.1000	1.0e-03	242	0.5716	0.5947	0.6992	0.6804
5e-04	0.1000	1e-04	17	0.4540	0.5956	0.7936	0.6951
5e-04	0.0000	0e+00	16	0.4330	0.5991	0.8069	0.6818

Table 10: Frozen SBERT, Weighted Cross Entropy, Sheldon vs. Leonard

learning_rate	dropout_rate	weight_decay	epoch	train_loss	val_loss	train_acc	val_acc
1e-04	0.2000	1.0e-03	183	0.4808	0.5773	0.7805	0.7012
5e-04	0.2000	1.0e-03	49	0.4801	0.5773	0.7809	0.7017
1e-04	0.2000	5e-04	101	0.4784	0.5782	0.7798	0.7008
5e-04	0.1000	1.0e-03	40	0.4866	0.5803	0.7778	0.6894
1e-04	0.1000	5e-04	97	0.4767	0.5813	0.7851	0.6913
1e-04	0.1000	1.0e-03	148	0.4985	0.5814	0.7672	0.6951
1e-04	0.0000	1.0e-03	182	0.4830	0.5814	0.7809	0.6984
5e-05	0.0000	1e-04	143	0.4542	0.5815	0.7987	0.7027
5e-05	0.2000	5e-04	204	0.4822	0.5820	0.7782	0.6951
5e-05	0.2000	1e-04	143	0.4713	0.5821	0.7854	0.6970
5e-05	0.0000	0e+00	115	0.4791	0.5822	0.7782	0.6922
5e-05	0.0000	5e-04	192	0.4778	0.5824	0.7818	0.6889
5e-05	0.1000	5e-04	194	0.4860	0.5824	0.7735	0.6899
5e-04	0.2000	5e-04	27	0.4615	0.5830	0.7899	0.6847
1e-04	0.0000	5e-04	110	0.4685	0.5830	0.7886	0.6965
5e-05	0.1000	1e-04	117	0.4942	0.5838	0.7649	0.6970
5e-04	0.1000	5e-04	27	0.4546	0.5839	0.7982	0.6951
5e-04	0.0000	5e-04	24	0.4673	0.5839	0.7901	0.7017
5e-05	0.2000	0e+00	120	0.4969	0.5840	0.7619	0.6932
5e-04	0.0000	1.0e-03	36	0.5004	0.5841	0.7635	0.6951
1e-05	0.0000	0e+00	503	0.5012	0.5843	0.7590	0.6903
1e-04	0.1000	0e+00	68	0.4662	0.5848	0.7850	0.6979
1e-05	0.2000	0e+00	510	0.5103	0.5854	0.7544	0.6884
1e-04	0.1000	1e-04	65	0.4839	0.5855	0.7737	0.6932
1e-05	0.1000	0e+00	466	0.5105	0.5855	0.7517	0.6927
1e-04	0.2000	0e+00	61	0.4895	0.5861	0.7706	0.6903
5e-05	0.1000	0e+00	117	0.4875	0.5861	0.7701	0.6889
1e-04	0.0000	0e+00	55	0.4880	0.5863	0.7691	0.6927
1e-04	0.0000	1e-04	68	0.4744	0.5863	0.7836	0.6932
1e-05	0.2000	1e-04	460	0.5243	0.5865	0.7397	0.6922
5e-05	0.0000	1.0e-03	241	0.5267	0.5866	0.7403	0.6880
1e-04	0.2000	1e-04	64	0.4918	0.5868	0.7667	0.7003
5e-05	0.2000	1.0e-03	183	0.5344	0.5874	0.7336	0.6870
1e-05	0.0000	1e-04	377	0.5292	0.5879	0.7352	0.6927
1e-05	0.2000	5e-04	526	0.5391	0.5883	0.7253	0.6903
5e-05	0.1000	1.0e-03	175	0.5389	0.5890	0.7286	0.6894
5e-04	0.2000	1e-04	21	0.4358	0.5895	0.8057	0.7055
1e-05	0.0000	5e-04	450	0.5424	0.5896	0.7226	0.6913
1e-05	0.1000	1e-04	336	0.5455	0.5906	0.7212	0.6884
1e-05	0.1000	5e-04	404	0.5533	0.5919	0.7137	0.6861
5e-04	0.2000	0e+00	17	0.4465	0.5931	0.7981	0.7045
5e-04	0.0000	1e-04	18	0.4423	0.5946	0.8046	0.6856
1e-05	0.2000	1.0e-03	335	0.5680	0.5953	0.7041	0.6823
1e-05	0.0000	1.0e-03	336	0.5670	0.5964	0.7039	0.6818
5e-04	0.1000	1e-04	20	0.4284	0.5970	0.8118	0.6861
5e-04	0.1000	0e+00	19	0.4132	0.5978	0.8210	0.6903
1e-05	0.1000	1.0e-03	209	0.5763	0.5985	0.6991	0.6809
5e-04	0.0000	0e+00	17	0.4243	0.6008	0.8133	0.6979

A.2 LoRA

Table 11: LoRA, Cross Entropy, 7-way classifier

rank	alpha	weight_decay	epoch	train_loss	val_loss	train_acc	val_acc
32	64	1e-04	10	1.5629	1.6189	0.4061	0.3821
8	16	0e+00	16	1.5650	1.6243	0.4038	0.3750
16	32	0e+00	13	1.5538	1.6273	0.4082	0.3750
16	32	1e-04	10	1.5901	1.6277	0.3940	0.3741
8	16	1e-04	12	1.5944	1.6303	0.3946	0.3699
32	64	0e+00	7	1.5943	1.6333	0.3954	0.3670

Table 12: LoRA, Weighted Cross Entropy, 7-way classifier

rank	alpha	weight_decay	epoch	train_loss	val_loss	train_acc	val_acc
16	32	0e+00	11	1.6833	1.7406	0.3443	0.3290
8	16	1e-04	10	1.7172	1.7422	0.3323	0.3135
32	64	1e-04	9	1.6836	1.7438	0.3438	0.3152
8	16	0e+00	10	1.7178	1.7465	0.3339	0.3026
16	32	1e-04	11	1.6899	1.7508	0.3462	0.3192
32	64	0e+00	6	1.7189	1.7530	0.3375	0.3138

Table 13: LoRA, Cross Entropy, Sheldon vs. Penny

rank	alpha	weight_decay	epoch	train_loss	val_loss	train_acc	val_acc
32	64	1e-04	6	0.4285	0.4520	0.7929	0.7859
8	16	1e-04	9	0.4374	0.4530	0.7886	0.7833
8	16	0e+00	12	0.4192	0.4534	0.7951	0.7818
16	32	0e+00	8	0.4264	0.4545	0.7917	0.7828
16	32	1e-04	9	0.4224	0.4567	0.7970	0.7781
32	64	0e+00	6	0.4272	0.4701	0.7944	0.7667

Table 14: LoRA, Weighted Cross Entropy, Sheldon vs. Penny

rank	alpha	weight_decay	epoch	train_loss	val_loss	train_acc	val_acc
16	32	0e+00	10	0.4233	0.4604	0.7905	0.7745
16	32	1e-04	9	0.4311	0.4664	0.7882	0.7818
8	16	0e+00	12	0.4253	0.4698	0.7887	0.7542
32	64	1e-04	6	0.4430	0.4726	0.7784	0.7385
8	16	1e-04	11	0.4373	0.4758	0.7840	0.7562
32	64	0e+00	8	0.4147	0.4769	0.7973	0.7625

Table 15: LoRA, Cross Entropy, Sheldon vs. Leonard

rank	alpha	weight_decay	epoch	train_loss	val_loss	train_acc	val_acc
8	16	1e-04	13	0.4929	0.5291	0.7512	0.7277
16	32	1e-04	11	0.4852	0.5320	0.7577	0.7249
32	64	0e+00	8	0.4843	0.5326	0.7562	0.7211
8	16	0e+00	10	0.5027	0.5337	0.7450	0.7183
32	64	1e-04	6	0.5074	0.5346	0.7444	0.7225
16	32	0e+00	13	0.4633	0.5421	0.7684	0.7301

Table 16: LoRA, Weighted Cross Entropy, Sheldon vs. Leonard

rank	alpha	weight_decay	epoch	train_loss	val_loss	train_acc	val_acc
16	32	0e+00	10	0.4905	0.5284	0.7500	0.7292
8	16	1e-04	12	0.4983	0.5286	0.7437	0.7367
16	32	1e-04	11	0.4880	0.5343	0.7558	0.7159
8	16	0e+00	6	0.5310	0.5365	0.7231	0.7268
32	64	1e-04	6	0.5112	0.5378	0.7364	0.7259
32	64	0e+00	7	0.4910	0.5430	0.7543	0.7202

A.3 Last-Layer

Table 17: Last-Layer, Cross Entropy, 7-way classifier

weight_decay	epoch	train_loss	val_loss	train_acc	val_acc
1e-04	6	1.5757	1.6287	0.3968	0.3726
0e+00	4	1.6036	1.6447	0.3863	0.3706

Table 18: Last-Layer, Weighted Cross Entropy, 7-way classifier

weight_decay	epoch	train_loss	val_loss	train_acc	val_acc
1e-04	6	1.6771	1.7650	0.3484	0.3210
0e+00	4	1.7177	1.7665	0.3298	0.3249

Table 19: Last-Layer, Cross Entropy, Sheldon vs. Penny

weight_decay	epoch	train_loss	val_loss	train_acc	val_acc
0e+00	7	0.4089	0.4642	0.7986	0.7745
1e-04	5	0.4437	0.4761	0.7799	0.7521

Table 20: Last-Layer, Weighted Cross Entropy, Sheldon vs. Penny

weight_decay	epoch	train_loss	val_loss	train_acc	val_acc
1e-04	7	0.4319	0.4736	0.7839	0.7615
0e+00	6	0.4293	0.5134	0.7876	0.6953

Table 21: Last-Layer, Cross Entropy, Sheldon vs. Leonard

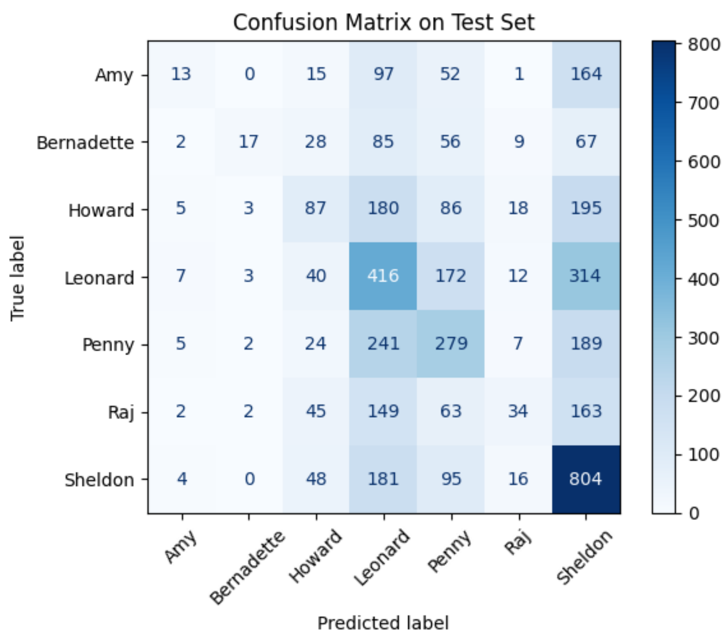
weight_decay	epoch	train_loss	val_loss	train_acc	val_acc
0e+00	5	0.5093	0.5433	0.7408	0.7154
1e-04	4	0.5325	0.5590	0.7220	0.7050

Table 22: Last-Layer, Weighted Cross Entropy, Sheldon vs. Leonard

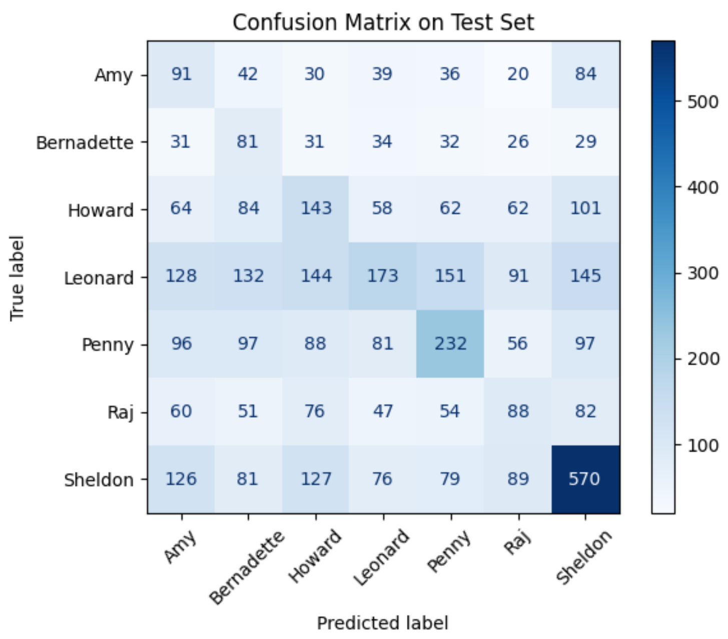
weight_decay	epoch	train_loss	val_loss	train_acc	val_acc
1e-04	4	0.5329	0.5507	0.7216	0.7135
0e+00	7	0.4801	0.5799	0.7603	0.6932

B Confusion matrices on test set

B.1 7-way classifier

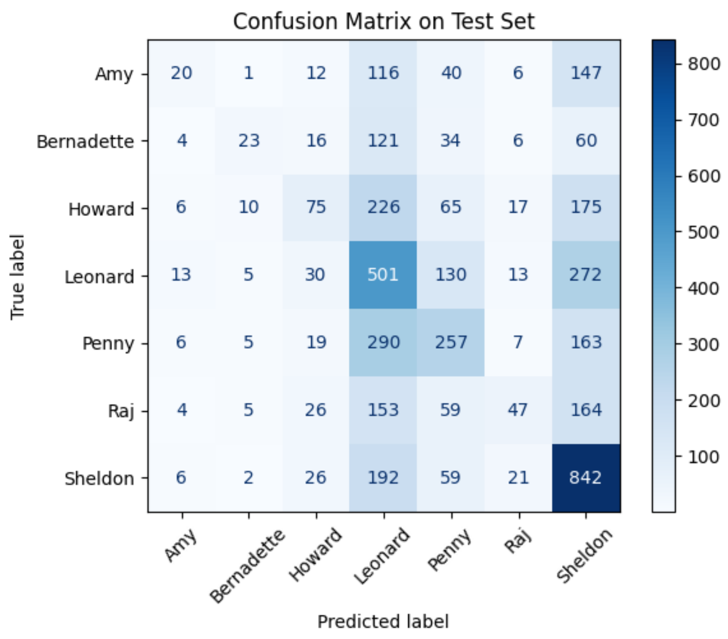


(a) Standard Cross-Entropy

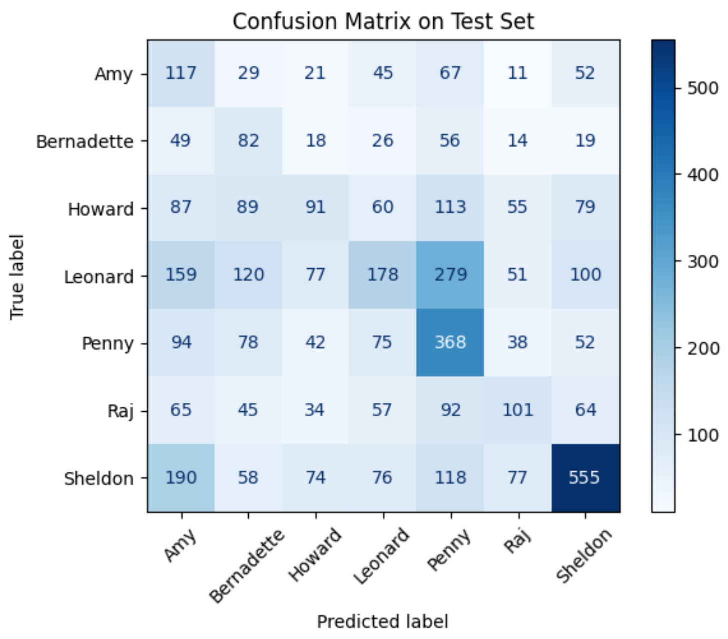


(b) Weighted Cross-Entropy

Figure 6: Confusion matrices for Frozen SBERT + MLP on 7-way classification task

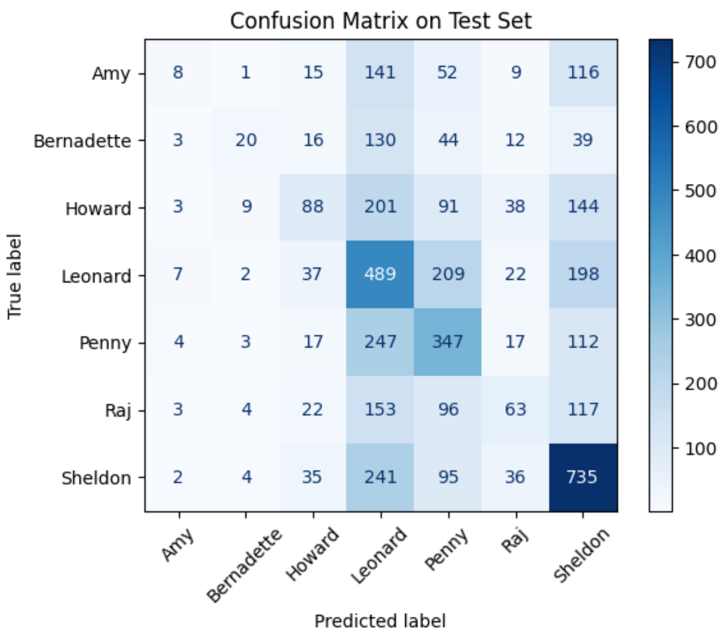


(a) Standard Cross-Entropy

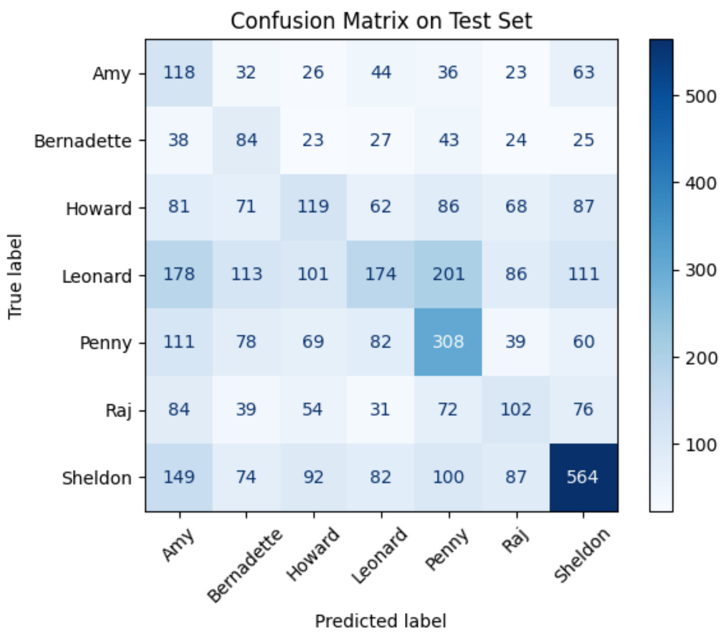


(b) Weighted Cross-Entropy

Figure 7: Confusion matrices for LoRA + MLP on 7-way classification task



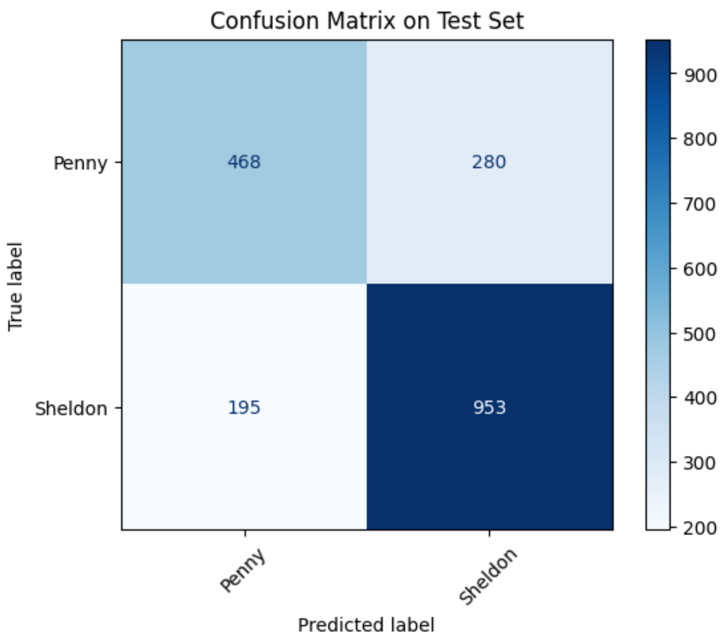
(a) Standard Cross-Entropy



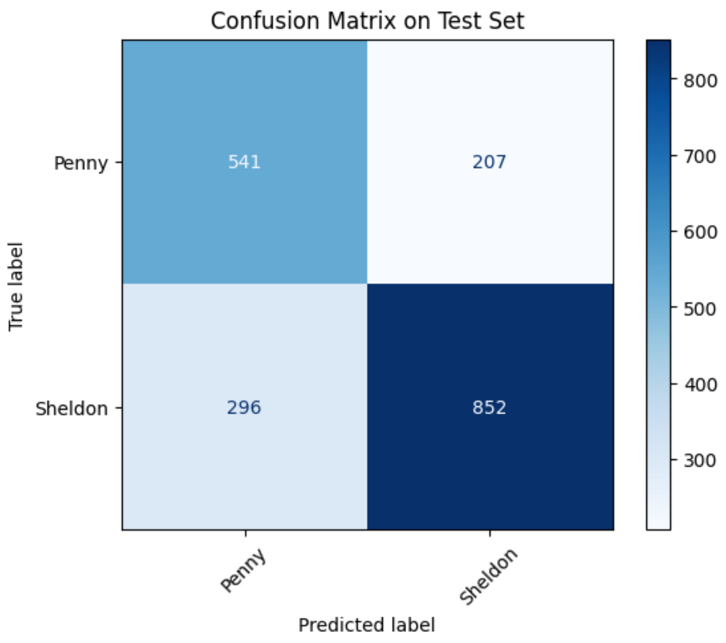
(b) Weighted Cross-Entropy

Figure 8: Confusion matrices for Last-Layer + MLP on 7-way classification task

B.2 Sheldon vs. Penny

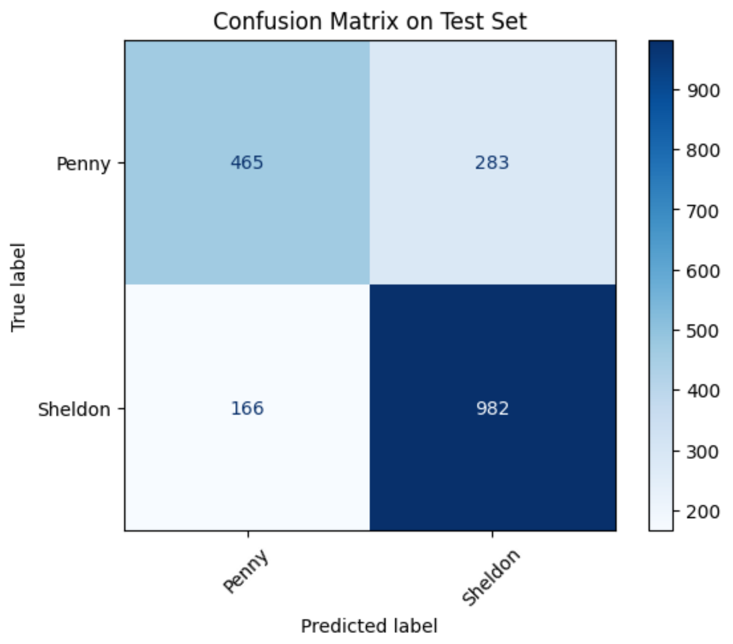


(a) Standard Cross-Entropy

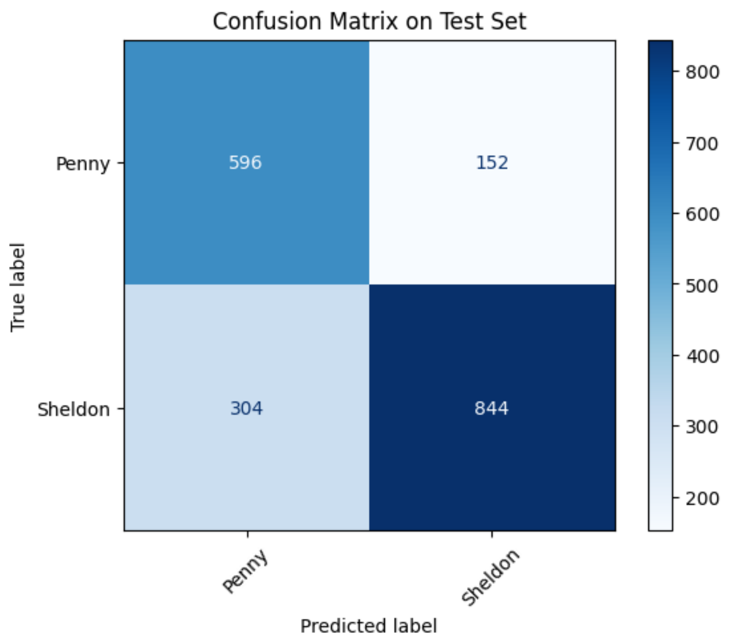


(b) Weighted Cross-Entropy

Figure 9: Confusion matrices for Frozen SBERT + MLP on Sheldon vs. Penny

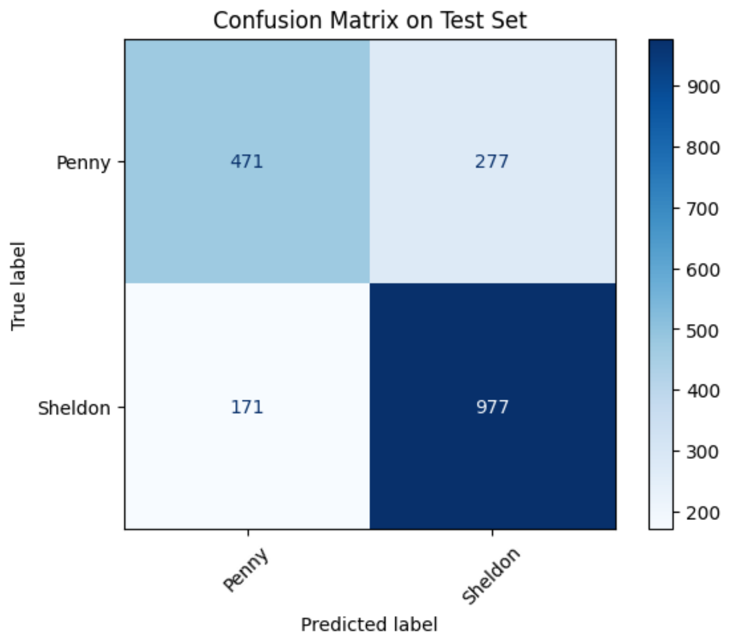


(a) Standard Cross-Entropy

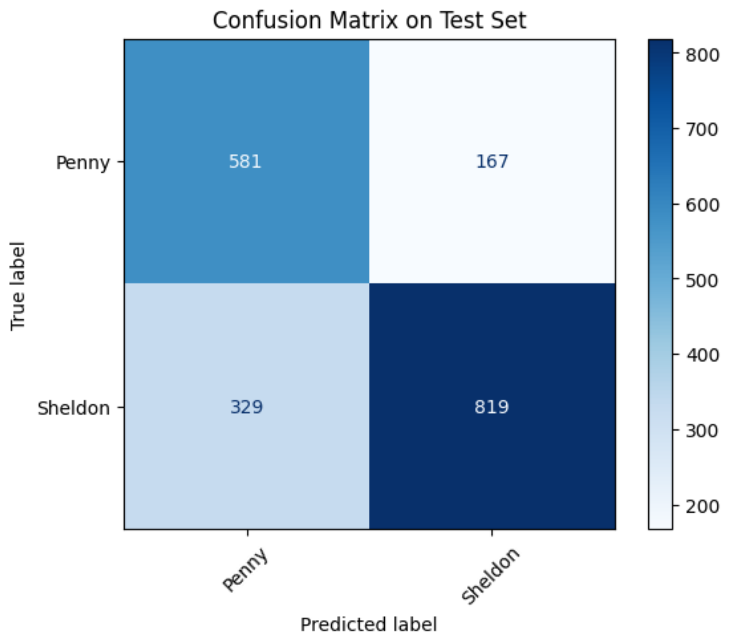


(b) Weighted Cross-Entropy

Figure 10: Confusion matrices for LoRA + MLP on Sheldon vs. Penny



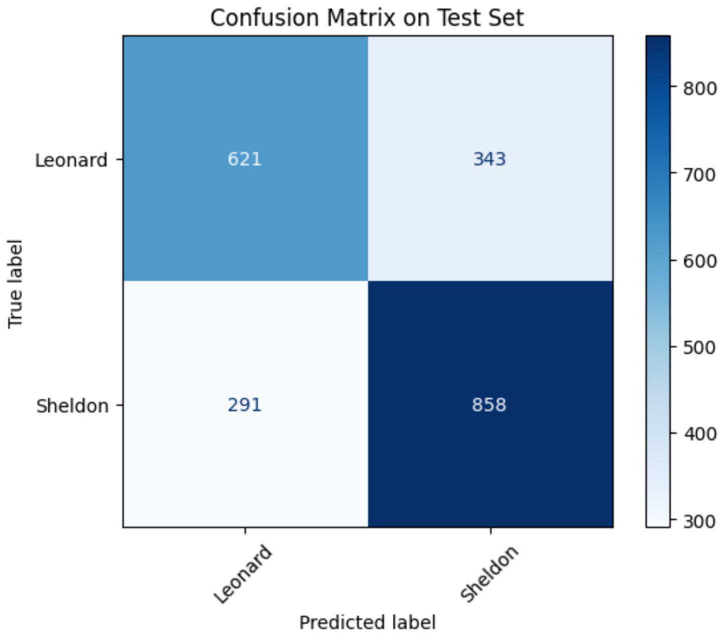
(a) Standard Cross-Entropy



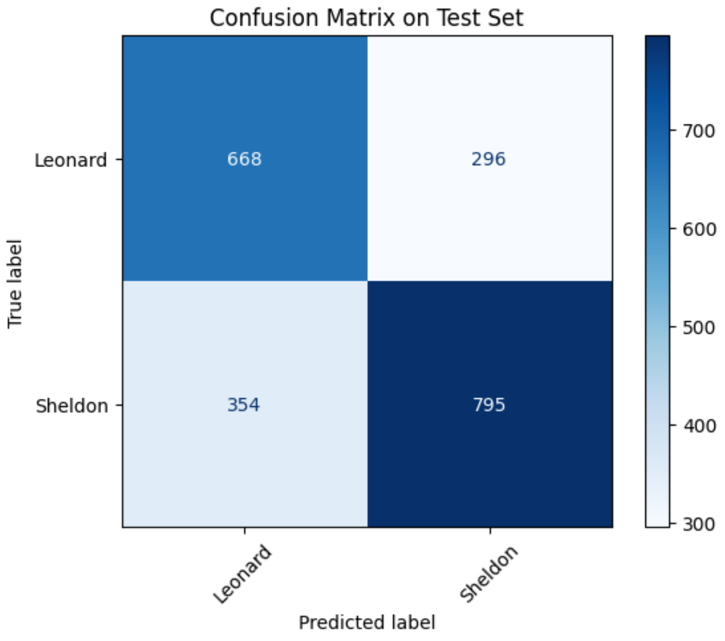
(b) Weighted Cross-Entropy

Figure 11: Confusion matrices for Last-Layer + MLP on Sheldon vs. Penny

B.3 Sheldon vs. Leonard

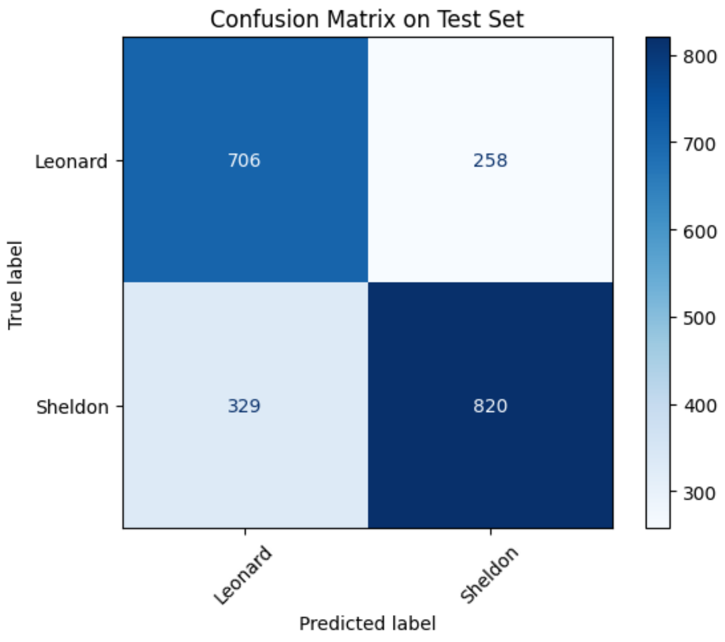


(a) Standard Cross-Entropy

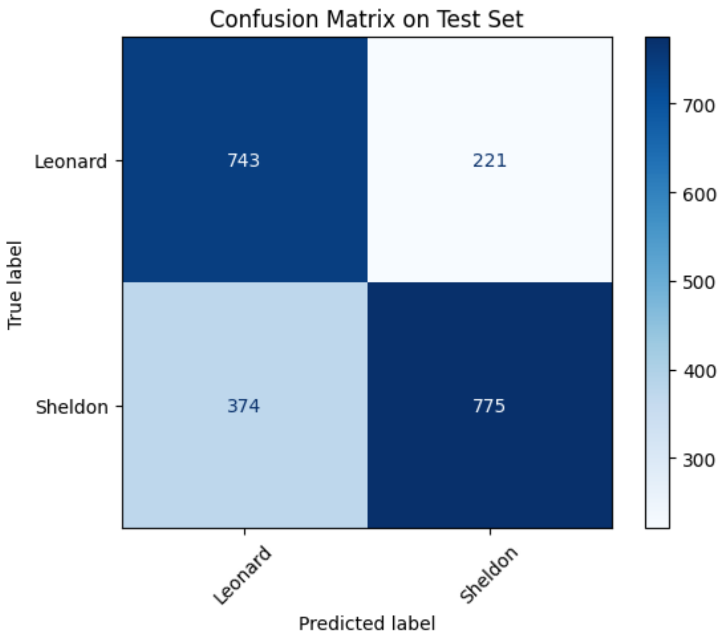


(b) Weighted Cross-Entropy

Figure 12: Confusion matrices for Frozen SBERT + MLP on Sheldon vs. Leonard

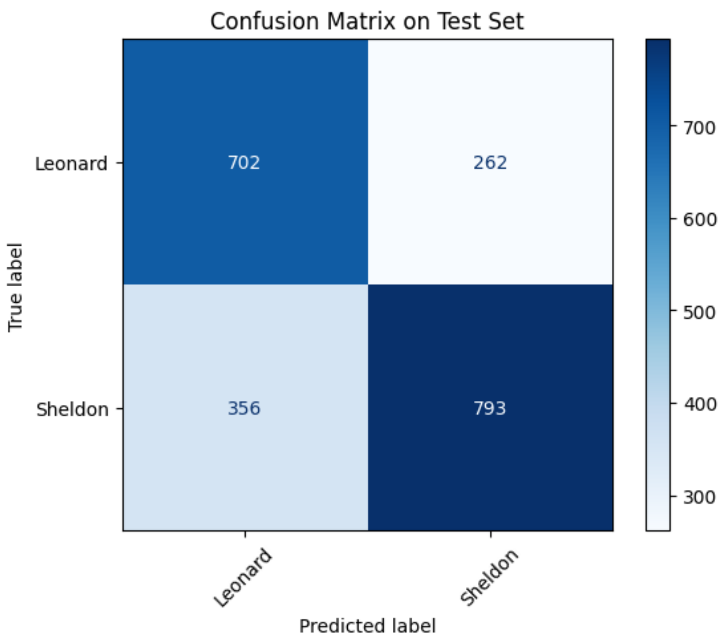


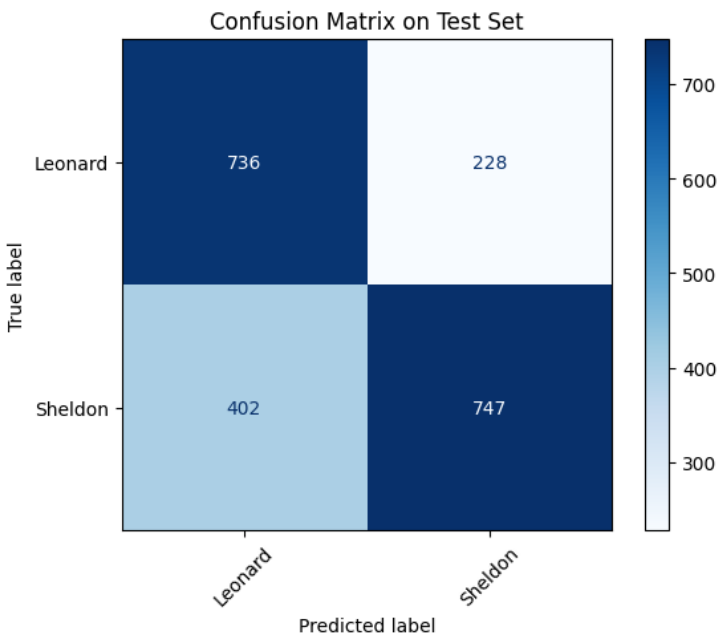
(a) Standard Cross-Entropy



(b) Weighted Cross-Entropy

Figure 13: Confusion matrices for LoRA + MLP on Sheldon vs. Leonard

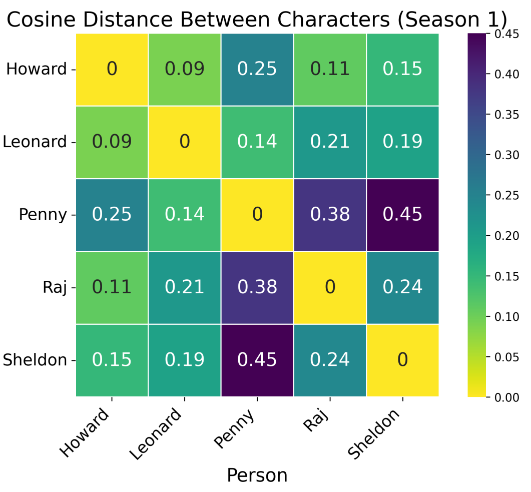




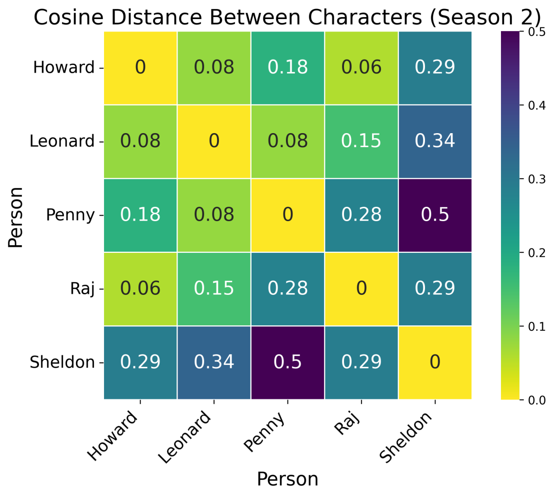
(a) Standard Cross-Entropy (b) Weighted Cross-Entropy

Figure 14: Confusion matrices for Last-Layer + MLP on Sheldon vs. Leonard

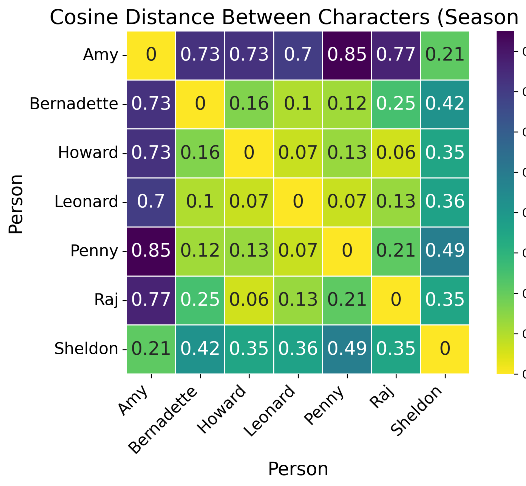
C Cosine distances per season between mean character embeddings



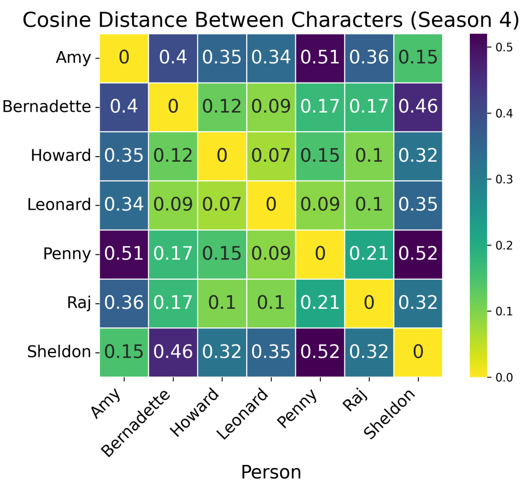
(a) Season 1



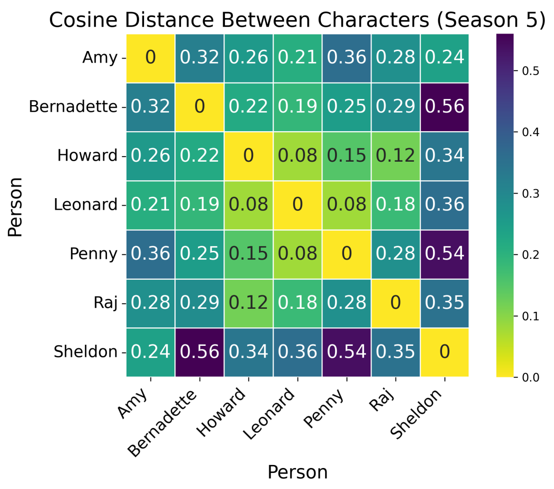
(b) Season 2



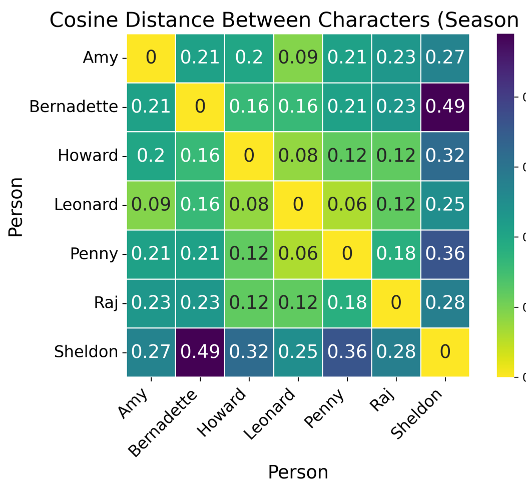
(c) Season 3



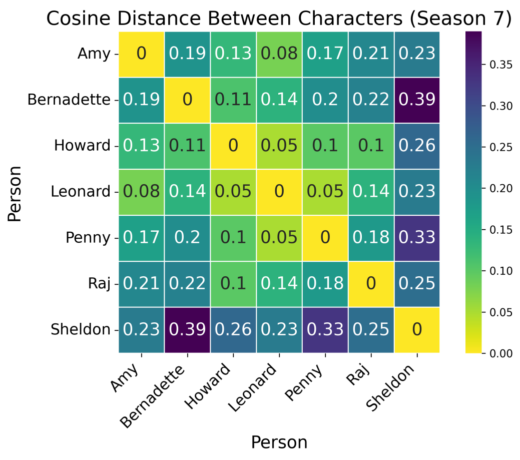
(d) Season 4



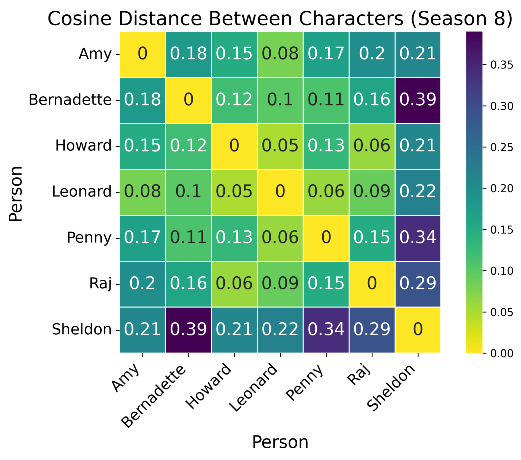
(e) Season 5



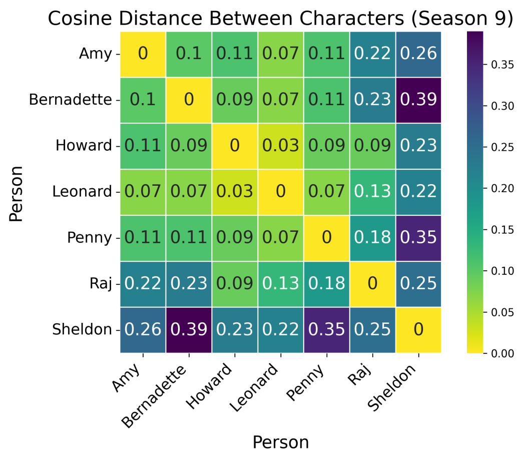
(f) Season 6



(g) Season 7

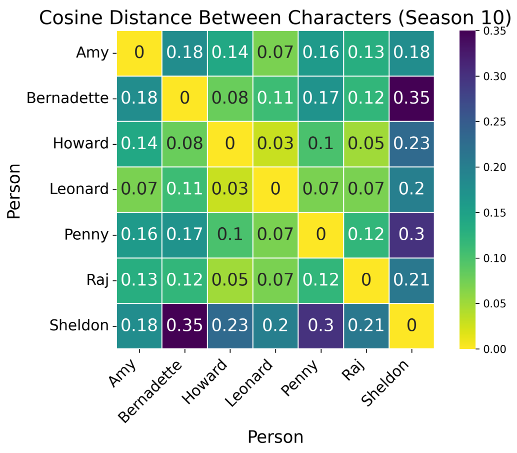


(h) Season 8



(i) Season 9

Figure 15: Cosine distances between characters — Seasons 1 to 9



	Amy	Bernadette	Howard	Leonard	Penny	Raj	Sheldon
Amy	0	0.18	0.14	0.07	0.16	0.13	0.18
Bernadette	0.18	0	0.08	0.11	0.17	0.12	0.35
Howard	0.14	0.08	0	0.03	0.1	0.05	0.23
Leonard	0.07	0.11	0.03	0	0.07	0.07	0.2
Penny	0.16	0.17	0.1	0.07	0	0.12	0.3
Raj	0.13	0.12	0.05	0.07	0.12	0	0.21
Sheldon	0.18	0.35	0.23	0.2	0.3	0.21	0

(a) Season 10

Figure 16: Cosine distances between characters — Season 10