Developing a Large Language Model-Based Playlist Continuation Recommender System

Enzo CHAROLOIS–PASQUA, Eléa VELLARD

Under the supervision of
Youssra REBBOUD, Pasquale LISENA and Raphael TRONCY

Semester project, January 2025

Table of Contents

Table of Contents

Introduction

Context:

- Music streaming platforms (e.g., Spotify) play a key role.
- Personalized recommendation is essential: automatically generating tailored playlists.
- This project builds upon the RecSys Challenge 2018 and the Million Playlist Dataset (MPD).

Overall Goal:

- Explore an up-to-date playlist generation approach using LLMs.
- Enhance playlist creation with semantic clustering, embeddings, and fine-tuning of a sentence-transformer model.

Table of Contents

Related Work

Recommender systems and music recommendation:

- Robin Burke, Alexander Felfernig, and Mehmet H Göker (2011). "Recommender systems: An overview". In: *Ai Magazine* 32.3, pp. 13–18
- Ching-Wei Chen et al. (2018). "Recsys challenge 2018: Automatic music playlist continuation". In: *Proceedings of the 12th ACM Conference on Recommender Systems*, pp. 527–528

Previous approaches:

- Diego Monti et al. (2018). "An ensemble approach of recurrent neural networks using pre-trained embeddings for playlist completion". In: *Proceedings of the ACM Recommender Systems Challenge 2018*, pp. 1–6

Technical assets:

- Nils Reimers and Iryna Gurevych (Nov. 2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, pp. 3982–3992
- Guy Shani and Asela Gunawardana (2011). "Evaluating Recommendation Systems". In: *Recommender Systems Handbook*. Ed. by Francesco Ricci et al. Boston, MA: Springer US, pp. 257–297

Table of Contents

Dataset

Spotify's Million Playlist Dataset

- 1,000,000 user-generated playlists.
- January 2010 - October 2017.
- 1,000 JSON slices.

Method Overview

Main Steps:

1. **MPD Preprocessing:** convert JSON to CSV, separating playlists/items/tracks.

Method Overview

Main Steps:

1. **MPD Preprocessing:** convert JSON to CSV, separating playlists/items/tracks.
2. **Semantic clustering:** embeddings and K-Means algorithm.

Method Overview

Main Steps:

1. **MPD Preprocessing:** convert JSON to CSV, separating playlists/items/tracks.
2. **Semantic clustering:** embeddings and K-Means algorithm.
3. **Fine-tuning a lightweight transformer:** classify playlists into thematic clusters.

Method Overview

Main Steps:

1. **MPD Preprocessing:** convert JSON to CSV, separating playlists/items/tracks.
2. **Semantic clustering:** embeddings and K-Means algorithm.
3. **Fine-tuning a lightweight transformer:** classify playlists into thematic clusters.
4. **Playlist Generation:** voting system.

Method Overview

Main Steps:

1. **MPD Preprocessing:** convert JSON to CSV, separating playlists/items/tracks.
2. **Semantic clustering:** embeddings and K-Means algorithm.
3. **Fine-tuning a lightweight transformer:** classify playlists into thematic clusters.
4. **Playlist Generation:** voting system.
5. **Evaluation:** Quantitative and qualitative measures.

Clustering: Embedding Model (1/3)

Idea: Identify semantic themes among playlists[1].

- FastText: lightweight but did not capture enough the contextual meaning of playlist titles.

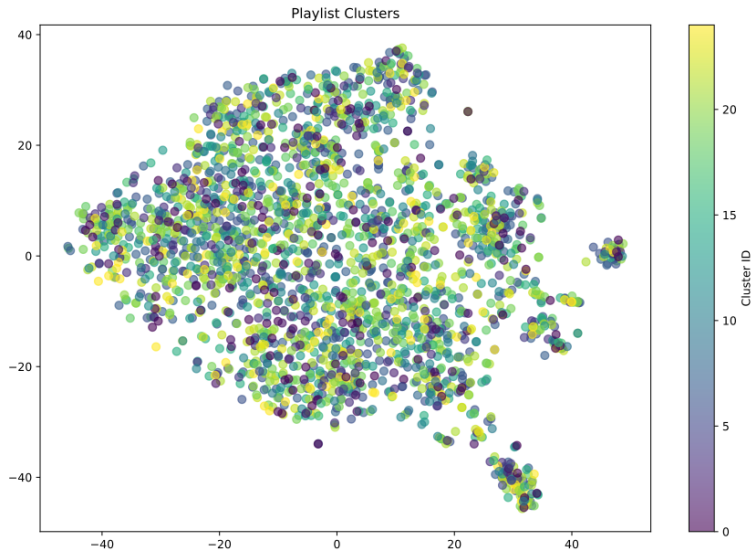

Figure 1: T-SNE plot of training data usign FastText

[1] Bohan Li et al. (2020). "On the sentence embeddings from pre-trained language models". In: *arXiv preprint arXiv:2011.05864*.

Clustering: Embedding Model (2/3)

- **BERT (Bidirectional Encoder Representations from Transformers):**
  - Uses self-attention mechanisms to learn contextual representations of tokens.
  - Not optimized to produce one fixed embedding for an entire sentence.
- **Sentence-BERT:**
  - Extends BERT with a *Siamese* architecture: two BERT modules process sentence pairs in parallel.[2]
  - Adds a pooling layer on top to produce fixed-size sentence embeddings.
  - Specially designed for sentence-level tasks.

Why Sentence-BERT for our project?

- More effective at capturing the semantic meaning of playlist titles.
- Efficient similarity comparison among a large set of sentences or playlists.

[2] Nils Reimers and Iryna Gurevych (Nov. 2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, pp. 3982–3992.

Clustering: Embedding Model (3/3)

1. **Model selection:** Sentence-BERT
2. **Content-based approach:** The playlist embedding is computed as the average of its track embeddings:

$$\mathbf{e}_{\text{playlist}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{e}_{\text{track}_i}$$

3. **Resulting Representation:** The resulting vector represents the overall semantic theme of the playlist.

Why Averaging?

- Averaging helps to smooth out noise and variability among individual tracks.
- It provides a stable, fixed-size representation for each playlist.

Clustering: Algorithm (1/2)

Why K-Means?

- A simple, well-known clustering algorithm.
- It partitions data into a predefined number of clusters by minimizing intra-cluster variance.

Why not DBScan:

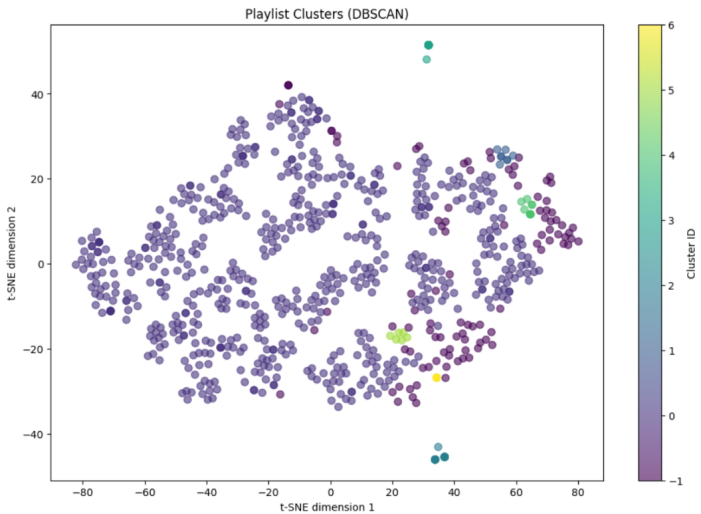

Figure 2: T-SNE plot of training data using DBScan

Clustering: Algorithm (2/2)

Choosing k:

- We empirically set $k = 200$ to maximize intra-cluster coherence.
- Other techniques (like the elbow method) were used to guide the choice, but our focus was on cluster coherence.

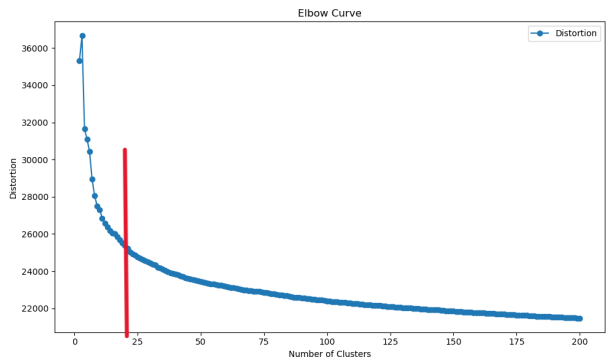

Figure 3: Elbow curve method. Suggesting an optimal $k \approx 22$.

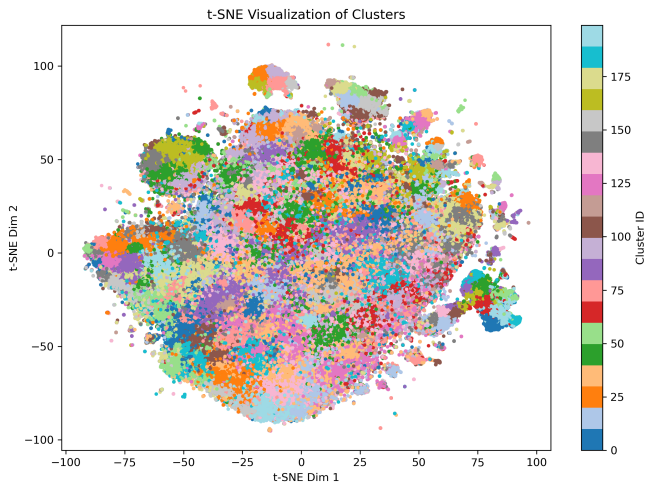

Figure 4: T-SNE plot for the whole dataset, using Sentence-BERT and K-Means algorithm.

Clustering: cleaning

Miscellaneous Clusters:

- Some clusters tend to be very large and less coherent.

Cleaning Strategy:

- **Post-Clustering Analysis:** Evaluate clusters for internal coherence. Simulate a manual check using the percentage of exact title matches.
- **Filtering:** Cleaning threshold $t = 2$.

Clustering: results

- **Cleaning results:** removed around 40% of playlists, of 21% clusters.
- **Final k:** $k = 158$
- **Split:** (80/10/10) split, ensuring a representation of each cluster in both sets.

Representation of each cluster in split

- So the model **learns** and is **evaluated** on the full diversity of playlists.
- To avoid **bias** towards clusters that might be overrepresented in one split.

Table of Contents

Fine-tuning the model: Goal

Idea: Adapt a pre-trained sentence-transformer model to better capture the semantic nuances of playlist titles,

- **Data:** Playlists titles
- **Labels:** Cluster IDs
- Using the **training set** to update its weights and adjust model parameters for improved thematic classification.
- Using the **validation set** to monitor performance and prevent overfitting.

Fine-tuning the model: Architecture (1/2)

Base Model: *all-MiniLM-L6-v2*

- A compact sentence Transformers, providing 384-dimensional embeddings[3][4][5].
- Adapted for classification.

Classification head:

- A dropout layer to prevent overfitting.
- A fully connected layer mapping the 384-dimension to the number of clusters.
- A mapping from original cluster IDs to sequential labels

[3] Omar Espejel (2022). "Train and Fine-Tune Sentence Transformers Models". In: *Hugging Face*.

[4] Yichu Zhou and Vivek Srikumar (2021). "A Closer Look at How Fine-tuning Changes BERT". In: *CoRR* abs/2106.14282. arXiv: 2106.14282. URL: https://arxiv.org/abs/2106.14282.

[5] Chi Sun et al. (2019). "How to fine-tune bert for text classification?" In: *Chinese computational linguistics: 18th China national conference, CCL 2019, Kunming, China, October 18–20, 2019, proceedings 18*. Springer, pp. 194–206.

Fine-tuning the model: Architecture (2/2)

Training Setup:

- **Loss Function:** Cross-entropy loss, standard for classification tasks.
- **Batch Size:** 8 (computation constraints).
- **Epochs:** 5 (computation constraints).
- **Learning Rate:** 2×10^{-5} to prevent overfitting (higher generalization accuracy).

Evaluation Strategy:

- Evaluation after each epoch.
- Monitoring both training and validation loss.
- Saving the model with the best validation performance.

Fine-tuning the model: Results

Observations:

- A clear decrease in loss.
- No signs of overfitting.

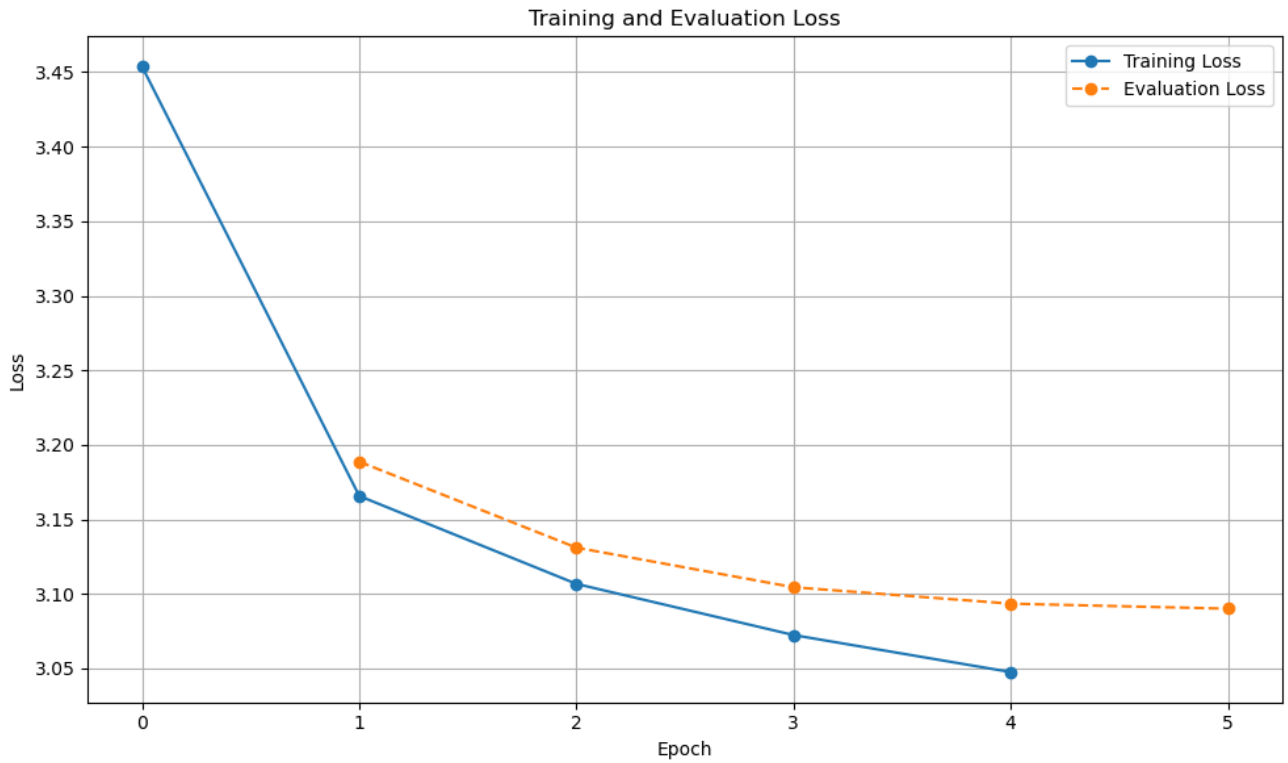

Figure 5: Training and validation loss across 5 epochs.

Table of Contents

Playlist Generation: Overview

Goal: Recommend tracks matching the semantic theme of a playlist title.

1. **Dataset embeddings:** Compute all playlists titles embeddings with the fine-tuned model.

[6] Tan Thongtan and Tanasanee Phienthrakul (July 2019). "Sentiment Classification Using Document Embeddings Trained with Cosine Similarity". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. Florence, Italy: Association for Computational Linguistics, pp. 407–414.

Playlist Generation: Overview

Goal: Recommend tracks matching the semantic theme of a playlist title.

1. **Dataset embeddings:** Compute all playlists titles embeddings with the fine-tuned model.
2. **User Input:** The user provides a playlist name.

[6] Tan Thongtan and Tanasanee Phienthrakul (July 2019). "Sentiment Classification Using Document Embeddings Trained with Cosine Similarity". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. Florence, Italy: Association for Computational Linguistics, pp. 407–414.

Playlist Generation: Overview

Goal: Recommend tracks matching the semantic theme of a playlist title.

1. **Dataset embeddings:** Compute all playlists titles embeddings with the fine-tuned model.
2. **User Input:** The user provides a playlist name.
3. **Input embedding:** The fine-tuned model converts the input title into a fixed-size embedding.

[6] Tan Thongtan and Tanasanee Phienthrakul (July 2019). "Sentiment Classification Using Document Embeddings Trained with Cosine Similarity". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. Florence, Italy: Association for Computational Linguistics, pp. 407–414.

Playlist Generation: Overview

Goal: Recommend tracks matching the semantic theme of a playlist title.

1. **Dataset embeddings:** Compute all playlists titles embeddings with the fine-tuned model.
2. **User Input:** The user provides a playlist name.
3. **Input embedding:** The fine-tuned model converts the input title into a fixed-size embedding.
4. **Similarity:** Compute cosine similarities[6] between the input embedding and embeddings of all existing playlists.

[6] Tan Thongtan and Tanasanee Phienthrakul (July 2019). "Sentiment Classification Using Document Embeddings Trained with Cosine Similarity". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. Florence, Italy: Association for Computational Linguistics, pp. 407–414.

Playlist Generation: Overview

Goal: Recommend tracks matching the semantic theme of a playlist title.

1. **Dataset embeddings:** Compute all playlists titles embeddings with the fine-tuned model.
2. **User Input:** The user provides a playlist name.
3. **Input embedding:** The fine-tuned model converts the input title into a fixed-size embedding.
4. **Similarity:** Compute cosine similarities[6] between the input embedding and embeddings of all existing playlists.

Why cosine similarity?

$$\text{cosine\_similarity}(\mathbf{e}_a, \mathbf{e}_b) = \frac{\mathbf{e}_a \cdot \mathbf{e}_b}{\|\mathbf{e}_a\| \|\mathbf{e}_b\|}$$

- It measures the **angle** between two vectors, giving a sense of their directional similarity rather than magnitude.
- Well-suited for comparing high-dimensional embeddings.

[6] Tan Thongtan and Tanasanee Phienthrakul (July 2019). "Sentiment Classification Using Document Embeddings Trained with Cosine Similarity". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. Florence, Italy: Association for Computational Linguistics, pp. 407–414.

Playlist Generation: Voting system

1. **Selecting Similar Playlists:** Identify k playlists with the highest cosine similarity scores.
2. **Track Extraction:** Extract all tracks from these similar playlists.
3. **Voting Mechanism:** The N most frequently occurring songs are recommended.

How to select k and N

- Empirically, ensuring a balance between sufficient amount of tracks and high similarity scores.
- $k = 50$ (need at least 13).
- $N = 66$ (average number of tracks per playlist).

Table of Contents

Results: Quantitative metrics

Quantitative Metrics[7]**:**

- **Precision@N:** Proportion of recommended tracks (S) that are relevant (R).

$$precision@N = \frac{len(R \cap S)}{len(S)} \tag{1}$$

- **Recall@N:** Proportion of relevant tracks that were successfully recommended.

$$recall@N = \frac{len(R \cap S)}{len(R)} \tag{2}$$

- **MRR@N (Mean Reciprocal Rank):** Focuses on the rank of the first relevant track.
  - A higher MRR indicates that the first correct recommendation appears earlier.

$$MRR@N = \frac{1}{\textit{Rank of the first relevant song in the recommended songs}} \tag{3}$$

[7] Guy Shani and Asela Gunawardana (2011). "Evaluating Recommendation Systems". In: *Recommender Systems Handbook*. Ed. by Francesco Ricci et al. Boston, MA: Springer US, pp. 257–297.

Results: Qualitative metrics

Qualitative Evaluation:

- Human assessment of thematic coherence and appeal.
- Evaluate each track to give the playlist an overall qualtitative score.
- Important because numerical metrics may not fully capture user satisfaction.

Examples

For a playlist named "Rock classics"

- Highway to Hell, AC/DC
- Smells Like Teen Spirit, Nirvana
- It's my life, Bon Jovi
- My heart will go on, Céline Dion

Qualitative score 75%

Results: Experiment

Experiment Setup:

- Using test set playlists.
- Final playlist: 66 tracks.
- Three models were compared:
  - **Fine-tuned sentence-transformers** (expected to capture refined semantics)
  - **Pre-trained sentence-transformers** (baseline)
  - **Llama3.1** (LLM in zero-shot mode) (see 29).
- Evaluated on a test set of 22 carefully selected playlists.

Playlists selection criteria

- $\geq$ 10 tracks
- Clear thematic alignment between playlist titles and their tracks.
- A mix of both niche (e.g., 'hawaii') and more broadly themed playlists (e.g., 'summer').

LLM Zero-shot vs. LLM Multi-shot

LLM Zero-shot:

- Uses only an instruction prompt without examples, in a simple format.
- *"Give me the 66 most relevant songs that were released before October 2017 for a playlist named input. Do a list of songs using the format (song name, artist), without any other text formatting."*
- Simpler and faster, but may lack precision.

LLM Multi-shot (using ChatGPT 4o):

- **First idea:**
  - Provide the model with the list of songs from our most similar playlists.
  - **Problem:** would add bias.
- **Adjustment:** Provides the model with the dataset to yield context-aware outputs.

Comparison

Showing no relevant improvement.

Results: Quantitative model comparison

Model	Fine-tuned model	Pre-trained model	llama3.1
Precision@10	0.0586	0.0586	0.0008
Recall@10	0.1295	0.1295	0.0119
MRR@10	0.2189	0.2189	0.0109

Table 1: Comparison of quantitative scores between 3 models.

Key Points:

- The fine-tuning did not yield an improvement over the pre-trained model (same model).
- **Hypothesis**: the weights were not correctly updated.
- Llama3.1's low scores suggest that our similarity-based voting approach is more relevant.

Results: Qualitative model comparison

Model	Fine-tuned model	Pre-trained model	llama3.1
Qualitative score	9.33	9.33	8.93

Table 2: Comparison of qualitative scores between 3 models.

Observations:

- Even when quantitative metrics were low, the recommended tracks often matched the theme.

Conclusion:

- Quantitative metrics alone do not capture the subjective quality of recommendations.
- Combining both evaluations provides a comprehensive view of system performance.

Table of Contents

Conclusion

Summary:

- Developed a pipeline combining semantic clustering, transformer fine-tuning, and a similarity-based voting system for playlist continuation.
- Leveraged Sentence-BERT to capture the semantic nuances of playlist titles.
- Compared multiple models (fine-tuned Sentence-BERT, pre-trained Sentence-BERT, and LLM zero-shot)

What did not work:

- Fine-tuning a model.
- Observing results of our whole pipeline.

Future Work

Next Steps:

- **Resolve the fine-tuning issue** to ensure that model weights are effectively updated.
- **Explore alternative clustering techniques** to reduce "miscellaneous" clusters.
- **Optimize** the recommendation pipeline (e.g., experiment with different hyperparameters in the voting system).
- **Incorporate user feedback** for better qualitative scores.

What We Learned

Key Takeaways:

- **NLP fundamentals:** especially embeddings and BERT-based models.
- **Fine-tuning a model:** Work with LLMs to tailor them for specific tasks.
- **Evaluating recommender systems:** Precision@N and Recall@N.
- **Research-oriented approach:** combined practical coding with theoretical approach.
- SSH server and GitHub for code storage.

Demo

Live demo to show the final recommender system

Choose a playlist title.

Thank You!

Do you have any questions?

Bibliography I

- Burke, Robin, Alexander Felfernig, and Mehmet H Göker (2011). "Recommender systems: An overview". In: *Ai Magazine* 32.3, pp. 13–18.
- Chen, Ching-Wei et al. (2018). "Recsys challenge 2018: Automatic music playlist continuation". In: *Proceedings of the 12th ACM Conference on Recommender Systems*, pp. 527–528.
- Espejel, Omar (2022). "Train and Fine-Tune Sentence Transformers Models". In: *Hugging Face*.
- Li, Bohan et al. (2020). "On the sentence embeddings from pre-trained language models". In: *arXiv preprint arXiv:2011.05864*.
- Monti, Diego et al. (2018). "An ensemble approach of recurrent neural networks using pre-trained embeddings for playlist completion". In: *Proceedings of the ACM Recommender Systems Challenge 2018*, pp. 1–6.

Bibliography II


Reimers, Nils and Iryna Gurevych (Nov. 2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, pp. 3982–3992.


Shani, Guy and Asela Gunawardana (2011). "Evaluating Recommendation Systems". In: *Recommender Systems Handbook*. Ed. by Francesco Ricci et al. Boston, MA: Springer US, pp. 257–297.


Sun, Chi et al. (2019). "How to fine-tune bert for text classification?" In: *Chinese computational linguistics: 18th China national conference, CCL 2019, Kunming, China, October 18–20, 2019, proceedings 18*. Springer, pp. 194–206.

Bibliography III


Thongtan, Tan and Tanasanee Phienthrakul (July 2019). "Sentiment Classification Using Document Embeddings Trained with Cosine Similarity". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. Florence, Italy: Association for Computational Linguistics, pp. 407–414.


Zhou, Yichu and Vivek Srikumar (2021). "A Closer Look at How Fine-tuning Changes BERT". In: *CoRR* abs/2106.14282. arXiv: 2106.14282. URL: https://arxiv.org/abs/2106.14282.