Reproducing: “Topic Modeling on Podcast Short-Text Metadata” by Valero et al.

YAHYA JABARY*, TU Wien, Austria
JOACHIM BIBERGER, TU Wien, Austria
SELINA REINHARD, TU Wien, Austria
NUSAIBA AHMED, TU Wien, Austria
JULIA CHALISSERY, TU Wien, Austria

In this study, we set out to reproduce the evaluation of the NEiCE algorithm as detailed by Valero et al. in their paper "Topic Modeling on Podcast Short-Text Metadata". Our focus was on replicating the results using the Deezer and iTunes datasets. We found that our reproduction of the Deezer dataset scores was fairly successful, albeit with some minor discrepancies. However, the reproduction of the iTunes dataset scores revealed significant differences from the original results. These findings suggest that while the NEiCE algorithm's performance on the Deezer dataset can be reliably reproduced, further investigation is needed to understand the variations observed in the iTunes dataset. Our study highlights the importance of reproducibility in research and the need for transparent reporting of methods and results to ensure the reliability of scientific findings.

ACM Reference Format:
Yahya Jabary, Joachim Biberger, Selina Reinhard, Nusaiba Ahmed, and Julia Chalissery. 2024. Reproducing: “Topic Modeling on Podcast Short-Text Metadata” by Valero et al.. 1, 1 (December 2024), 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

In this paper, we aim to reproduce the evaluation of the NEiCE algorithm as presented by Valero et al. [3].

1 Introduction

Named Entity informed Corpus Embedding (NEiCE), is a topic modeling algorithm that uses named entities (NE) to improve the quality of topics extracted from short-text content. The algorithm was introduced in the paper “Topic Modeling on Podcast Short-Text Metadata” by Valero et al. [3] from Deezer Research. It is based on the CluWords [4] algorithm, which clusters words based on their nearest neighbors. The authors claim that NeICE outperforms other topic modeling algorithms of the same class, such as Non-negative matrix factorization (NMF), Short-text topic modeling via non-negative matrix factorization (SeaNMF) [2] and Clustering words (CluWords) [4], on podcast metadata from Deezer, Spotify and iTunes.

The motivation behind this study stems from the nature of podcast metadata, which typically consists of short text such as titles and descriptions. Traditional topic modeling algorithms often struggle with short text due to the lack of context and sparse data, even when concatenated into pseudo-documents. However, NMF-based algorithms have shown promise in handling short text more effectively compared to probabilistic models, such as the Generalized Polya Urna Dirichlet Multinomial Mixture (GPU-DMM) [1]. Additionally, NMF-based algorithms offer better interpretability than

Authors' Contact Information: Yahya Jabary, jabaryyahya@gmail.com, TU Wien, Austria; Joachim Biberger, TU Wien, Austria; Selina Reinhard, TU Wien, Austria; Nusaiba Ahmed, TU Wien, Austria; Julia Chalissery, TU Wien, Austria.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
Manuscript submitted to ACM

neural models, such as the Negative sampling and Quantization Topic Model (NQTM) [5]. NEiCE offers an improvement over CluWords by leveraging named entities, which are more informative and coherent than regular words.

1.1 NEiCE Algorithm

The NEiCE algorithm consists of a (1) preprocessing stage and a (2) topic modeling stage. The preprocessing stage prepares the podcast metadata for topic modeling by enriching the text with named entities, generating contextualized word embeddings and creating document-level representations. The topic modeling stage then applies Non-negative Matrix Factorization (NMF) to discover relevant topics from the preprocessed data.

1.1.1 Preprocessing Stage. The preprocessing stage begins with named entity recognition (NER) in both the title and description fields of podcast metadata. The system employs the Radboud Entity Linker (REL) library for linking these entities to Wikipedia entries. Within this linking process, the Flair library, which utilizes embeddings, performs Named Entity Recognition (NER) to detect entity mentions. It then uses wikipedia2vec to identify unique candidates from a list of potential matches.

For each identified named entity, the system generates a Wikipedia page reference along with a confidence score. This score is crucial as it determines how the entity will be treated in subsequent processing – either as a simple text span, as a confirmed named entity or as individual words requiring separate processing.

The vocabulary cleaning phase incorporates the NameDataset library to eliminate overly common named entities such as actors and athletes that might not contribute meaningfully to topic identification. Following the cleaning process, the system applies a specialized NE-related re-weighting to the term frequency (tf) factor.

The preprocessing stage handles different datasets with specific criteria. For both iTunes and Spotify datasets, the system removes duplicate titles and eliminates entries where the combined title and description contain fewer than three terms. The Spotify dataset undergoes additional language filtering, utilizing both fastText and CLD3 to ensure only English-language podcasts are retained. The Deezer dataset, which is the largest among the three, contains similar metadata including creator-provided information, titles, descriptions and show names in English. Across all datasets, genres with fewer than 300 shows are excluded to ensure robust topic modeling.

1.1.2 Topic Modeling Stage. The Topic Modeling Stage uses Non-negative Matrix Factorization (NMF) as the primary technique. It begins with the creation of a document-term matrix X, where each row represents a podcast and each column represents a term from the vocabulary. The matrix X is then factorized into two non-negative matrices W and H, such that $X \approx WH$. In this factorization, W represents the document-topic matrix, while H represents the topic-term matrix.

To enhance the topic modeling process, the authors propose the novel Entity-informed Contextual Embedding (NEiCE). This representation leverages named entities, which are often present in podcast metadata, to improve the quality of discovered topics. It combines contextual word embeddings with named entity information, allowing for a more nuanced and accurate topic modeling process.

The optimization problem for NMF is formulated as minimizing the Frobenius norm of the difference between X and WH, subject to non-negativity constraints on W and H. This is expressed mathematically as $\min(W, H \geq 0) \cdot \|X - WH\|^2 F$. The resulting factorization provides a low-rank approximation of the original document-term matrix, with each column of W representing a topic and each row of H representing the importance of terms within that topic.

1.2 Evaluation

The evaluation primarily uses C_V topic coherence, which correlates best with human judgment of topic ranking. The C_V score is calculated as:

$$CV(k) = \frac{1}{T}\sum_{i=1}^{T}\cos(v_{NPMI}(t_i), v_{NPMI}(t_{1:T}))$$

where T is the number of top words per topic, NPMI vectors are computed using Wikipedia as the external corpus and the final coherence is averaged across all K topics.

The hyperparameters specified in the original paper are as follows:

- Number of top words (T): { 10 }
- Number of topics (K): { 20, 50, 100, 200 }
- REL confidence threshold: 0.9 for named entity linking
- α^{word} range: { 0.2, 0.3, 0.4, 0.5 }
- α^{ent} range: { 0.3, 0.4 }

Table 1 shows the topic coherence scores C_V obtained by NEiCE for each (α^{word},α^{ent}) configuration on the Deezer, Spotify and iTunes datasets. We want to reproduce this table and conduct a statistical analysis to evaluate our confidence in the authors’ results.

Dataset	Deezer				Spotify				iTunes			
K	20	50	100	200	20	50	100	200	20	50	100	200
NEiCE (0.2, 0.3)	50.2	48.9	51.4	48.4	51.7	49.0	45.2	46.5	49.3	43.3	49.5	47.0
NEiCE (0.2, 0.4)	53.1	49.2	50.8	50.6	48.7	48.7	43.5	41.7	47.2	49.5	50.7	51.3
NEiCE (0.3, 0.3)	48.5	52.1	51.5	49.8	52.2	49.0	47.5	47.6	50.3	52.5	49.0	48.2
NEiCE (0.3, 0.4)	53.3	50.9	55.3	51.6	50.1	48.5	51.1	49.8	52.5	49.5	49.2	49.8
NEiCE (0.4, 0.3)	53.2	51.5	52.2	50.0	53.2	49.5	50.5	45.9	52.8	50.1	50.6	51.1
NEiCE (0.4, 0.4)	56.4	52.6	48.1	49.0	51.0	48.2	47.3	47.8	52.4	51.9	49.9	47.4
NEiCE (0.5, 0.3)	52.5	56.3	50.8	55.4	51.3	47.7	45.6	45.4	50.6	46.5	46.7	49.0
NEiCE (0.5, 0.4)	56.3	60.6	54.9	53.3	55.0	49.9	46.7	45.0	50.5	52.0	48.7	46.1

Table 1. Topic coherence scores C_V (in %) obtained by NEiCE for each (α^{word},α^{ent}) configuration on the Deezer, Spotify and iTunes datasets as reported by Valero et al. [3] (Table 5).

2 Strategies

Our approach to reproducing the results involves closely following the steps outlined in the original paper and making reasonable decisions when encountering ambiguities or issues. Afterward, we will perform a statistical analysis to assess our confidence in the reproduced results.

Specifically, the following assumptions on hyperparameters and configurations had to be made:

- Number of topics: 10, 20, 50, 100 (common NMF arguments)
- Number of neighbors: 5, 10, 20, 500 (common CluWords arguments)

Fortunately, the authors have made their source code available on GitHub[1], which saved us from having to write any code from scratch. They did not implement the other baseline algorithms but provided links to the original papers. Due

[1]https://github.com/deezer/podcast-topic-modeling

to resource constraints, we decided to focus solely on reproducing the NEiCE results, as presented in Table 5 from the original paper.

The README provides clear instructions on how to reproduce the results. Each file is well-commented and there are two Docker environments: one for training and one for evaluation. Links to the datasets and library binaries are provided. To reproduce the results, one must set up the environment, follow the instructions and apply the hyperparameters as specified in the paper. However, the authors note that due to updated dependencies (REL, flairNLP), the vocabulary has changed and exact scores may not be reproducible. We expect distribution of scores to be similar.

All experiments were conducted on a MacBook Pro with an Apple M2 Pro chip and 16 GB of memory, running macOS 14.6.1 (23G93).

3 Difficulties

In this section, we discuss the difficulties we encountered while following the instructions provided by the authors.

3.1 Unresolvable Issues

Some difficulties we faced were not resolvable.

Spotify Dataset Unavailability. The Spotify dataset has been unavailable since December 2023[2]. This means we were constrained to the Deezer and iTunes datasets.

Non-Byte-Aligned Weights. The weights in `enwiki_20180420_300d.pkl` are not byte-aligned, which can cause inaccuracies and segfaults when used with PyTorch or other common libraries. This means we can not guarantee the correctness of the results obtained using these weights.

Missing Dependency Lock renders Container Unusable. The provided Docker container lacks a compiled or locked dependency file, making it near impossible to ensure dependency version compatibility. The REL dependency, cloned via git, does not specify a commit tag, which adds to the complexity since it relies on various other libraries. We managed to work around this by checking the open-source commit history and selecting a commit from early 2022, around the time the paper was published. However, this did not resolve all issues. A similar issue was encountered with the NER model from Flair, which requires a specific commit hash from the Huggingface repository. This was not specified in the container, so we had to manually find the correct commit hash by checking the date. However, a dependency mismatch in `entity_linking/radboud_entity_linker_batch.py` caused cryptic SQLite errors when calling `MentionDetection`, which we were unable to patch. This ultimately rendered the provided Docker container unusable, after spending several days manually shotgun-debugging dependency mismatches.

Misconfigured GPU Container. We were unable to leverage the GPU with the provided Docker container, both on our local machine and on Google Colab. The latter was due to the cgroup configuration bug, which prevents the use of Docker containers with GPU support. We had to rewrite the container configuration to use the CPU-only implementation. This significantly increased the time required for evaluation. This was further exacerbated by the slow evaluation loop, with each cycle taking 100-600 seconds and the entire evaluation process taking around 2-3 days in total to complete.

[2]https://podcastsdataset.byspotify.com/

Non-portable Container Configuration. The provided Docker container was not portable across different architectures. We had to set the arch emulation flag to make it work on ARM64 machines. Specifically, the `gcld3` library does not build on ARM64, even if the `protobuf` dependency is installed.
Ultimately this meant not having any information, neither about the Python version nor the dependency versions used in the original paper. This in itself vastly reduces our chances of achieving the same results as reported by the authors.

3.2 Resolved Issues

To ensure reproducibility, we created a Docker Compose YAML file that specifies the exact operating system and runtime versions. After a successful build, we also automatically dump the PIP environment to a `requirements.txt` file.
Additionally, to ensure reproducibility, we set seeds for all common libraries such as `random`, `numpy`, `torch` and `os` at the beginning of every file in the source code. This way, a single iteration is sufficient for evaluation. After resolving all environment-related issues by using the latest versions of the dependencies and letting PIP handle the version resolution, we successfully executed the code.
We observed that many functions in the codebase were deprecated due to missing backward compatibility of the dependencies. Consequently, we had to manually rewrite parts of the code to ensure compatibility with the latest library versions.
Specifically, we had to replace the deprecated "df.append(d, ignore_index=True)" with "pd.concat([df, pd.DataFrame([d])], ignore_index=True)" in "main_prepro.py" since the former is deprecated and disabled. We also had to replace "get_feature_names" with "get_feature_names_out" to resolve an "AttributeError" with "CountVectorizer" in "data_preprocessing/utils.py" and "neice_model/wikipedia2vec_model.py". Additionally, we had to replace the sklearn "NMF" argument "alpha" with "alpha_W" to resolve a non-existent parameter error in "neice_model.py". Finally, we had to wrap the flair NER model with the "SequenceTagger" wrapper and use the weights from the Feb 26, 2021 commit on huggingface[3] to resolve the "AttributeError" with "CountVectorizer" in "main_prepro.py". Another minor issue was that the path in "data_preprocessing/utils.py" was relative to the script (not the working directory) and used Linux path separators. We had to replace the path with an absolute path and use the correct path separators.

4 Key Findings

We ran statistical analyses on the reproduced results to compare them with the original results.

4.1 Deezer Scores Analysis

The t-test for Deezer Scores yielded a statistic of 0.2283 and a p-value of 0.8194. This high p-value (> 0.05) suggests that there is no statistically significant difference between the actual and expected Deezer Scores. We fail to reject the null hypothesis. In simpler terms, it means that we successfully reproduced the Deezer Scores.
The 95% confidence intervals for actual (51.9947, 52.3533) and expected (51.8943, 52.3832) Deezer Scores overlap considerably. This overlap further supports the conclusion that there is no significant difference between the actual and expected scores.

[3]Commit Hash: 3d3d35790f78a00ef319939b9004209d1d05f788

The actual variance (4.2554) is lower than the expected variance (7.9090) for Deezer Scores. This suggests that the reproduced results have less spread than the original data, but it doesn't necessarily indicate a problem with the reproduction.
The Kolmogorov-Smirnov (KS) Test for Deezer Scores resulted in a statistic of 0.2035 and a very low p-value (1.1304e-09). This low p-value suggests that there is a statistically significant difference in the distributions of the actual and expected scores, despite the similar means indicated by the t-test.
The reproduction of Deezer Scores appears to be relatively successful. While the means are not significantly different (as shown by the t-test), there is some evidence of distributional differences (from the KS-test). This suggests that the central tendency of the scores was reproduced well, but there might be some differences in the shape or spread of the distributions. Figure 2 further validates this conclusion by showing the distribution of expected and actual Deezer Scores.

4.2 iTunes Scores Analysis

The t-test for iTunes Scores produced a statistic of 14.4127 and an extremely low p-value (5.3178e-43). This indicates a significant difference between the actual and expected iTunes Scores. The reproduction attempt for iTunes Scores appears to deviate significantly from the original results. In other words, the null hypothesis is rejected and the actual and expected scores are not equal.
The 95% confidence intervals for actual (51.1179, 51.4063) and expected (49.3459, 49.7190) iTunes Scores do not overlap at all.
The actual variance (2.7515) is lower than the expected variance (4.6078) for iTunes Scores. Similar to the Deezer Scores, the reproduced results show less spread than the original data.
The KS-test for iTunes Scores yielded a statistic of 0.4521 and an extremely low p-value (2.0552e-47). This further confirms a significant difference in the distributions of the actual and expected iTunes Scores.
The reproduction of iTunes Scores shows significant deviations from the original results. Both the t-test and KS-test indicate substantial differences between the actual and expected scores. The confidence intervals do not overlap and the means are significantly different. This suggests that the reproduction attempt for iTunes Scores was not successful. Figure 3 visualizes this discrepancy.

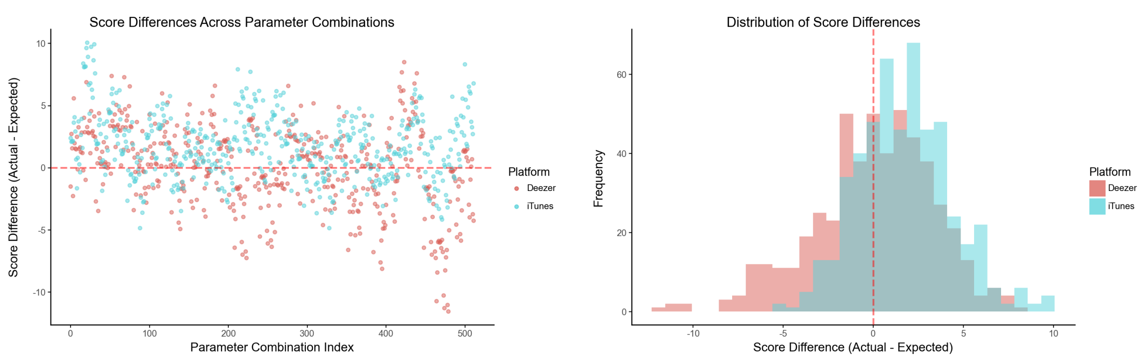

Fig. 1. Distribution of differences between our reproduced C_V scores and the authors' reported scores.

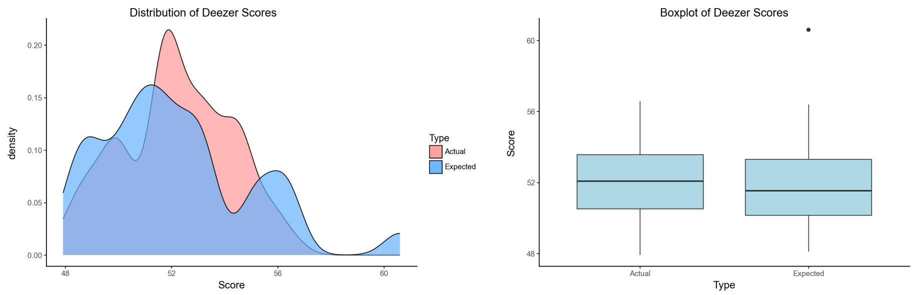

Fig. 2. Distribution of expected and actual C_V scores for the Deezer dataset.

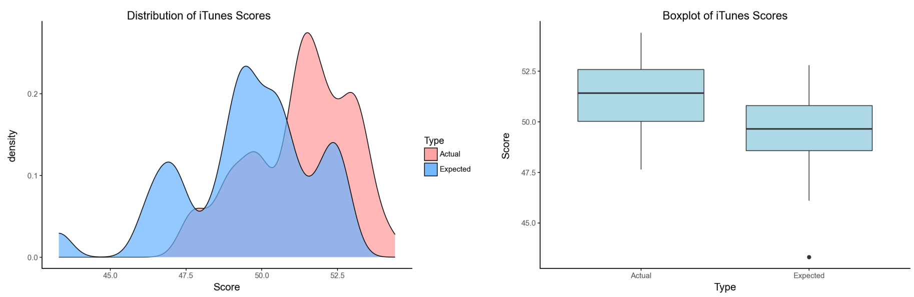

Fig. 3. Distribution of expected and actual C_V scores for the iTunes dataset.

One general observation that applies to both Deezer and iTunes Scores is that the actual variances are consistently lower than the expected variances. This pattern suggests that the reproduced results have less spread than the original data. While this could indicate a systematic difference in how the scores were calculated or collected in the reproduction attempt, it could also be due to the smaller sample size of the reproduced data compared to the original data.
It’s also worth noting that we sampled 512 scores in total, which is a relatively large, when compared to the actual scores provided by the authors. This in itself could have introduced some bias in our results.

5 Conclusion

In this paper, we attempted to reproduce the evaluation of the NEiCE algorithm as presented by Valero et al. [3]. We encountered several difficulties, including unresolvable issues such as the unavailability of the Spotify dataset, non-byte-aligned weights and missing dependency locks. We also faced resolved issues, such as deprecated functions, missing dependency versions and non-portable container configurations. Despite these challenges, we successfully executed the code and conducted a statistical analysis to compare our reproduced results with the original results.
In conclusion, while the Deezer Scores reproduction seems reasonably successful with some caveats, the iTunes Scores reproduction shows significant differences from the original results. Further investigation into the methodology, data collection process and calculation methods for iTunes Scores would be warranted to understand the source of these discrepancies.

References

[1] Chenliang Li, Haoran Wang, Zhiqian Zhang, Aixin Sun, and Zongyang Ma. 2016. Topic Modeling for Short Texts with Auxiliary Word Embeddings. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Pisa, Italy) *(SIGIR '16)*. Association for Computing Machinery, New York, NY, USA, 165–174. https://doi.org/10.1145/2911451.2911499

[2] Tian Shi, Kyeongpil Kang, Jaegul Choo, and Chandan K Reddy. 2018. Short-text topic modeling via non-negative matrix factorization enriched with local word-context correlations. In *Proceedings of the 2018 world wide web conference*. 1105–1114.

[3] Francisco B Valero, Marion Baranes, and Elena V Epure. 2022. Topic modeling on podcast short-text metadata. In *European Conference on Information Retrieval*. Springer, 472–486.

[4] Felipe Viegas, Sérgio Canuto, Christian Gomes, Washington Luiz, Thierson Rosa, Sabir Ribas, Leonardo Rocha, and Marcos André Gonçalves. 2019. CluWords: Exploiting Semantic Word Clustering Representation for Enhanced Topic Modeling. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining* (Melbourne VIC, Australia) *(WSDM '19)*. Association for Computing Machinery, New York, NY, USA, 753–761. https://doi.org/10.1145/3289600.3291032

[5] Xiaobao Wu, Chunping Li, Yan Zhu, and Yishu Miao. 2020. Short text topic modeling with topic distribution quantization and negative sampling decoder. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1772–1782.