Quotes Recommender System, Final Technical Report

Semester Project, Practical Machine Learning and Deep Learning, Fall 2023, Innopolis University

Team

Full name	Innopolis e-mail	Individual contribution
Vladimir Makharev	v.makharev@innopolis.university	Data collection and preprocessing, rectools methods, experiments, demo, presentation
Artem Batalov	a.batalov@innopolis.university	Data preprocessing, methods, experiments, demo
Georgii Budnik	g.budnik@innopolis.university	Method, prompt engineering, presentation

GitHub Repository

https://github.com/kilimanj4r0/quotes-recsys

Streamlit Demo

https://huggingface.co/spaces/batalovme/quotes-recsys

Project Description

"If I quote others, it is only to better express my own thought"
— Michel de Montaigne is a French philosopher of the Renaissance

Inspirational quotes for everyday help a person achieve goals and believe in themselves, do not give up and move on. The goal of our project is to develop a system of recommendations for quotes based on the answer to the question "How was your day?" to lift and improve the user's mood for the next day. The recommendation engine will be based on NLP mechanisms and deep learning.

Tags

RecSys, NLP, ML, DL, Feature Extraction, Text Classification, LLMs

Work Done

Data, models, tools exploration

Our Initial Thoughts

1. Find labeled datasets of diary-style texts and quotes, then somehow play with their labels.
2. Fine just datasets of texts that are close to the diary-style domain and quotes datasets (maybe parse from somewhere), then label all of these manually using one multi-label classifier.

Found datasets

1. go_emotions
2. sem_eval_2018_task_1
3. journal-entries-with-labelled-emotions
4. Diary-Entry-To-Rap
5. Quotes Dataset

We experimented with 1 and 2 labeled datasets to match labels from the 5th dataset. We took only texts of diaries from 3 and 4 datasets and concatenated them into one. So, as a result we made experiments on concatenated **Diaries dataset and Quotes dataset**.

Found classifiers from the top of HuggingFace Trending Models

1. `roberta-base-go_emotions` — 27 labels + Neutral label
2. `twitter-roberta-base-emotion-multilabel-latest` — 11 labels (or 4 labels from tweetnlp [1])
3. `bert-base-uncased-emotion` — 6 labels (Ekman emotions)
4. `emotion_text_classifier` — 6 labels (Ekman emotions)
5. `EmoRoBERTa` — 27 labels + Neutral label

We experimented with the **first two classifiers**. There are reasons why we declined 3, 4, and 5:

- `roberta-base-go_emotions` outperforms `EmoRoBERTa` on the same dataset
- `bert-base-uncased-emotion` and `emotion_text_classifier` are trained to predict only 6 Ekman emotions, so we consider it as not enough for

our task. Also, according to Demszky *et al.* [2], the 6 emotion categories proposed by Ekman in 1992 are very basic and recent findings in psychology offer a more complex "semantic space" of emotion.

Tools

We used pandas for data processing, HuggingFace and PyTorch for experiments libraries mostly. We also used RecSys models and metrics from the most recent and new open-source library RecTools released by MTS [3].

Task understanding and setting

The goal is to build a content-based recommendation system. By choosing this approach, we will recommend to user *x* similar to previous items rated highly by *x*.

We need to construct a dataset of user interactions with items. Our setting is:

- **user**: represented by unique id and text (diary-style) + features of the text (emotions)
- **item**: represented by unique id and quote + features of the quote (emotions)
- **interaction**: a value that denotes whether user liked the item or not, i.e., it can be either -1 or 1

Therefore, by combining all things we will get our utility matrix (dataset of interactions).

Inspired by and based on Habr article about RecTools [4] and lecture about Recommendation Systems from the course we came up with the solution steps:

1. Solve Gathering Ratings problem
    1. Collect samples that represent **user** and **item** (filtering suitable data)
    2. Construct feature vector for each **user** and **item** by using multi-label classifier (choosing somehow appropriate classifier)
    3. Using a baseline model, create a dataset of "reasonable" pairs (diary-style text, quote) so that we can annotate it.
    4. To solve a cold start problem, build a dataset manually (or using external tools like OpenAI API) by creating **interaction** examples based on the dataset of pairs (diary-style text, quote).
    5. Split the dataset into train and test.
2. Solve Extrapolating Utilities problem
    1. Write a model that will be training embeddings using CosineEmbeddingsLoss.
    2. Train different RecSys models
3. Evaluate methods
    1. Evaluate (and compare) trained models by well-known RecSys metrics.

Data preparation

We worked with Diaries dataset (1648 samples, it is our **users**) and Quotes dataset (852 samples, it is our **items**).

Labeling of datasets

We attempted to make labels of quotes and diaries unified using different approaches (overlapping labels, computing most frequent ones, etc.). So, we kept classes from classifiers and tried to cast classes from the Quotes dataset to them. Files with these datasets are:

- `quotes_manual_labeled_reddit.csv`
- `quotes_manual_labeled_twitter.csv`

However, we came up with just simple labeling of Diaries and Quotes dataset using two classifiers: `roberta-base-go_emotions` and `twitter-roberta-base-emotion-multilabel-latest`. It was inference of these classifiers, so we get out diaries and quotes text labeled by 28 and 11 classes (emotions) respectively, where each class is represented by a score between 0 and 1. Files with these datasets are:

- `diaries_labeled_reddit.csv`
- `diaries_labeled_twitter.csv`
- `quotes_labeled_reddit.csv`
- `quotes_labeled_twitter.csv`

Dataset of “reasonable” pairs of diary and quote

Utilizing a baseline model (details in the next section), we created a dataset of diary and quote pairs. Files with these datasets are:

- `diaries_quotes_emb_reddit.csv`
- `diaries_quotes_emb_twitter.csv`

Generation of interactions dataset

We wanted to solve the cold start problem and assumed that annotating the dataset of diary and quote pairs with LLMs will be beneficial. We used OpenAI API to make inference on `gpt-4` and `gpt-3.5-turbo` with our prompts.
Files with these datasets are:

- **`diaries_quotes_emb_twitter_interactions.csv`**
  Consists of 519 samples, was created by `gpt-4` with the system prompt `“You are an expert recommendation evaluator. Now you must evaluate recommended quote of famous person for the person diary note. Just write '1' if quote is suitable, and '-1' if unsuitable"` and the instruction prompt `"Text: {Text}\n\nQuote: {e.Quote}”`.

- **`diaries_quotes_emb_reddit_interactions.csv`**

  Consists of 544 samples, was created by `gpt-3.5-turbo` with the prompt engineered system prompt `“Imagine you are the primary evaluator of the recommendation system. System recommend quote to the diary text by user (user answers to the question \"How was your day?\"). The system have cold start problem, that is why we ask you to solve a specific task (will be announced in the next prompt). You have a dataset of pairs (diary text, quote text). The diary texts were collected journal entries from people reflecting on their day. The quote was selected by baseline model that uses cosine similarity distance between embeddings from multi-label (28 emotions) classifier on reddit texts.”` and instruction prompt `“Your task is to evaluate each pair of diary text and quote text how much (in your opinion) the quote fits the corresponding diary, giving a rating of "1" - if it fits enough or "-1" - if it is not fits enough. Make decisions based on how the quote could lift the mood of the user and mood of most people who would be shown this quote for a similar text. Write ONLY "1" or "-1"\n\nDiary text: {e.Text}\nQuote text: {e.Quote}”`.

(Additional) Converting data into RecTools format

We constructed a RecTools format Dataset. It involves making feature vectors out of emotion label scores for user and item and assuming timestamps with 1 minute difference between each interaction. Interaction value was converted to 2 and 0.5 (from 1 and -1) since it should be a weight. Resulting table is (`user_id, item_id, weight, datetime, user_features, item_features`).

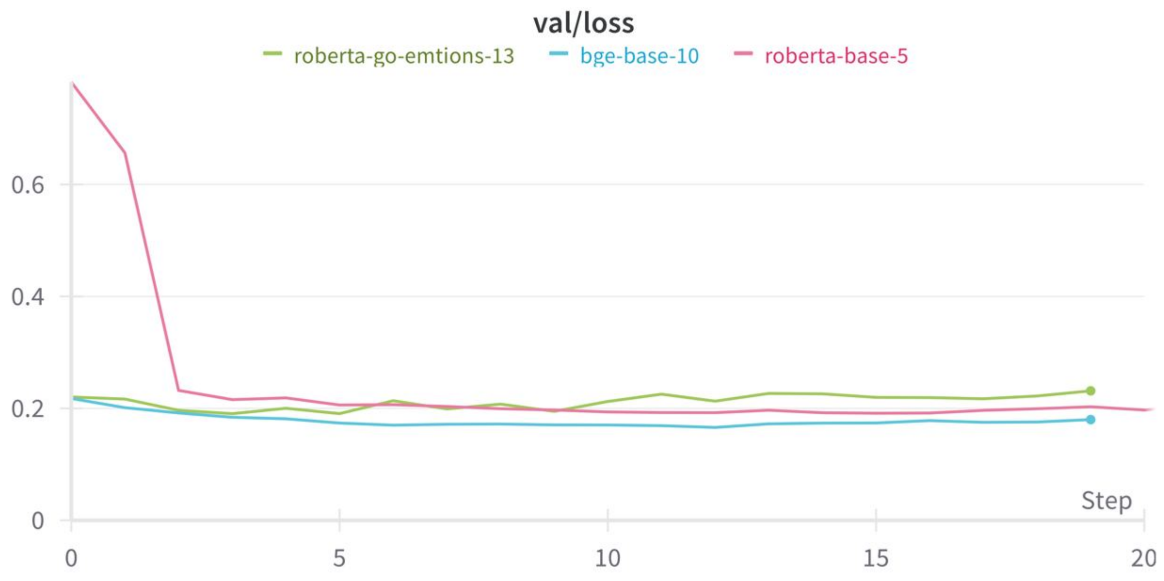
Models building and training

Baseline

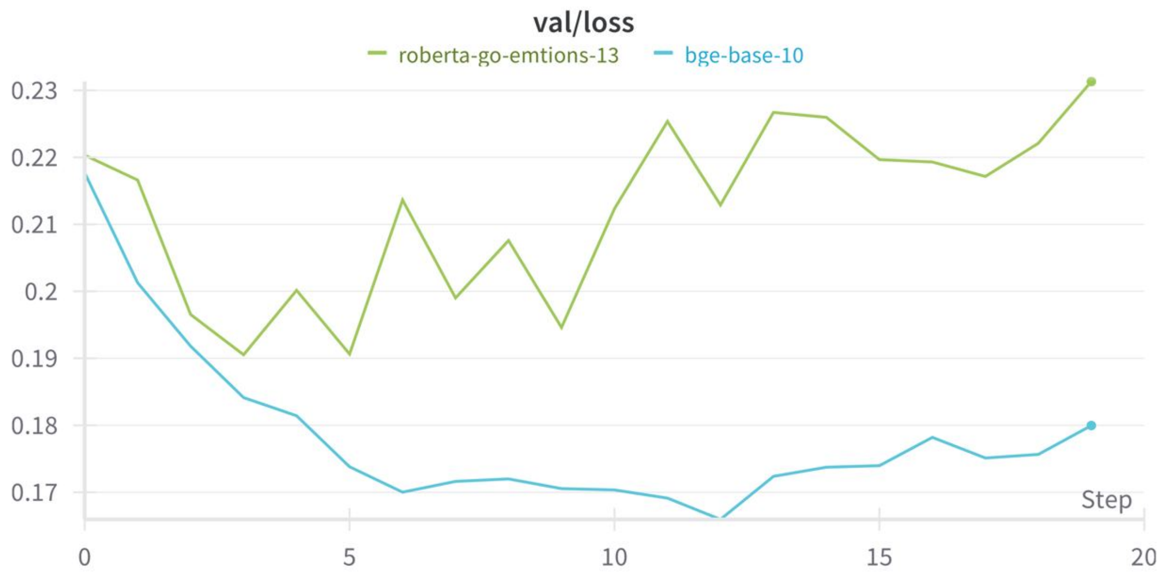
We made baseline models to recommend quotes based on cosine similarity when we have no historical data. Utilizing embeddings that are learned by chosen classifiers (available at HuggingFace) `roberta-base-go_emotions` and `twitter-roberta-base-emotion-multilabel-latest`. We can compare embeddings of diary and quote texts. To get embeddings for text and quote we turned the last classification layer from both models. We also used embedding models such as roberta-base (chosen to compare with embeddings from classifiers) and BAAI/bge-base-en-v1.5 (chosen as best performing according to MTEB leaderboard [5]) to create other baselines. We used cosine similarity between embeddings to get the distance score. So, we selected randomly from quotes whose distance is above `threshold=0.8` and the top-1 quote if no quotes’ distances are above the `threshold=0.8` for each diary. We used random to add the exploration component to our baseline model.

Embeddings fine-tuning with CosineEmbeddingLoss

We proposed a method to fine-tune embedding models with CosineEmbeddingLoss to let models learn to recommend better quotes for diaries. The models used in the baseline were fine-tuned. These models are learned on our assumed and generated interactions dataset. The validation loss curves are:



Zooming into (without `roberta-base`) validation loss curves are:



The validation loss is unstable. Therefore, there might be two problems: 1) the dataset has a small number of samples, 2) the dataset has poor data.

(Additional) RecTools models

We made a recommendation tables (`user_id, item_id, score, rank`) by using different models, namely:

1. PopularModel (Popularity-Based): Recommends top-k popular items for each user based on overall popularity.
2. RandomModel (Random Recommendation): Provides random recommendations by selecting k items randomly for each user.
3. PureSVDModel (Matrix Factorization): Utilizes Singular Value Decomposition for matrix factorization to recommend top-k items.

The choice of `k=10` ensures a consistent evaluation metric, representing the number of recommendations each model provides for users in the recommendation tables.

Models evaluation

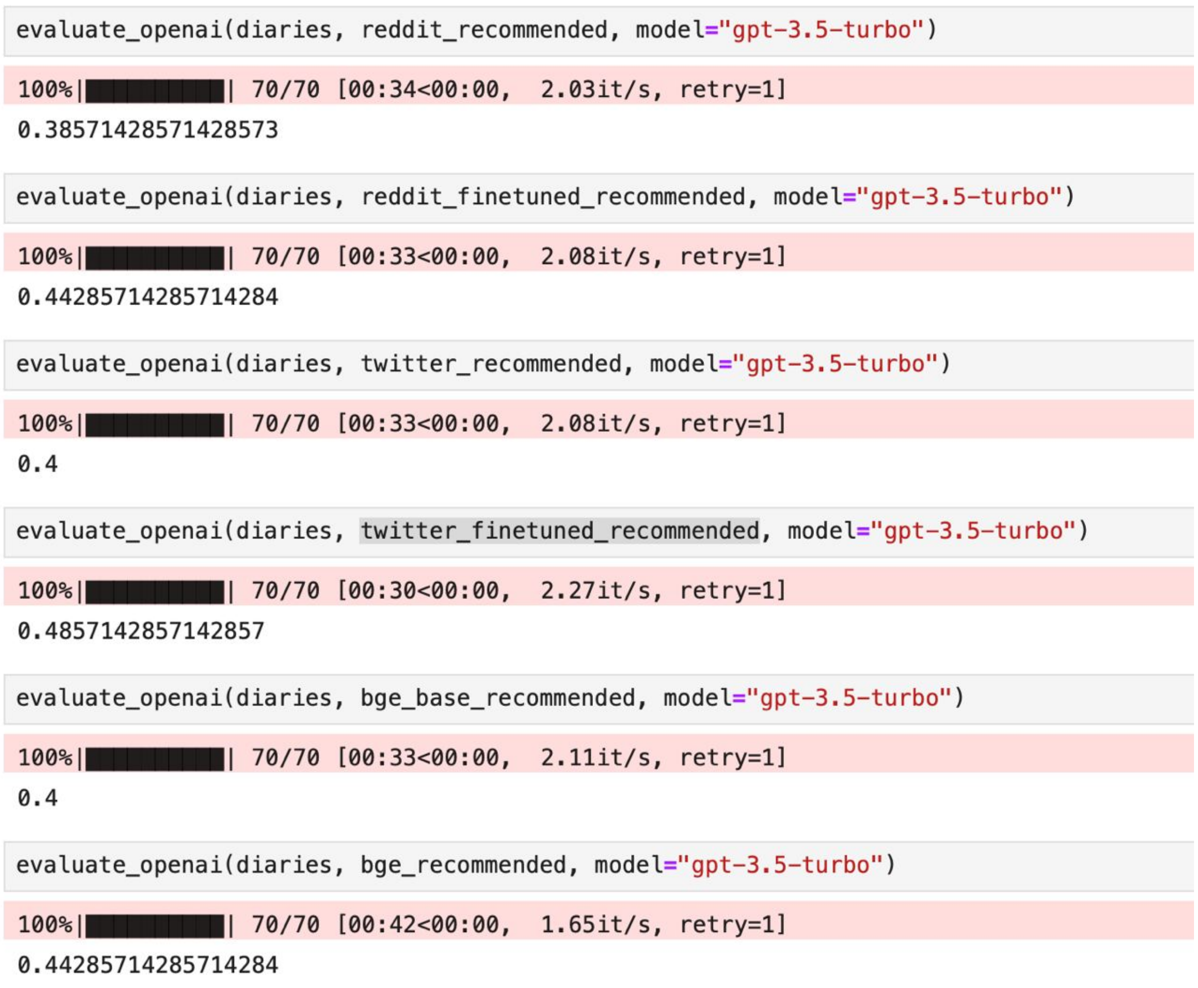
We introduce our own `Custom-GPT-Score` metric. This metric represents accuracy, i.e., number of positive (equal to 1) interactions out of all samples, where interaction value is generated by GPT; prompt is the same as last engineered in the generation of interaction dataset section of data preparation.
The interactions dataset was splitted into train and validation sets. And also we have a test set to evaluate the model on a `Custom-GPT-Score`.

Baseline

The baseline models were evaluated on a test set using `Custom-GPT-Score` with `gpt-3.5-turbo` model, consisting of 70 samples.

Embeddings fine-tuning with CosineEmbeddingLoss

Each model was evaluated using CosineEmbeddingLoss before (i.e., baseline) and after fine-tuning of embeddings on the validation set. We tried to understand how our models were fine-tuned on the new knowledge (see previous “Models building and training” section). The evaluation results on the test set of fine-tuned models are represented by computed `Custom-GPT-Score`:



```
evaluate_openai(diaries, reddit_recommended, model="gpt-3.5-turbo")
```

```
100%|██████████| 70/70 [00:34<00:00,  2.03it/s, retry=1]
0.38571428571428573
```

```
evaluate_openai(diaries, reddit_finetuned_recommended, model="gpt-3.5-turbo")
```

```
100%|██████████| 70/70 [00:33<00:00,  2.08it/s, retry=1]
0.44285714285714284
```

```
evaluate_openai(diaries, twitter_recommended, model="gpt-3.5-turbo")
```

```
100%|██████████| 70/70 [00:33<00:00,  2.08it/s, retry=1]
0.4
```

```
evaluate_openai(diaries, twitter_finetuned_recommended, model="gpt-3.5-turbo")
```

```
100%|██████████| 70/70 [00:30<00:00,  2.27it/s, retry=1]
0.4857142857142857
```

```
evaluate_openai(diaries, bge_base_recommended, model="gpt-3.5-turbo")
```

```
100%|██████████| 70/70 [00:33<00:00,  2.11it/s, retry=1]
0.4
```

```
evaluate_openai(diaries, bge_recommended, model="gpt-3.5-turbo")
```

```
100%|██████████| 70/70 [00:42<00:00,  1.65it/s, retry=1]
0.44285714285714284
```

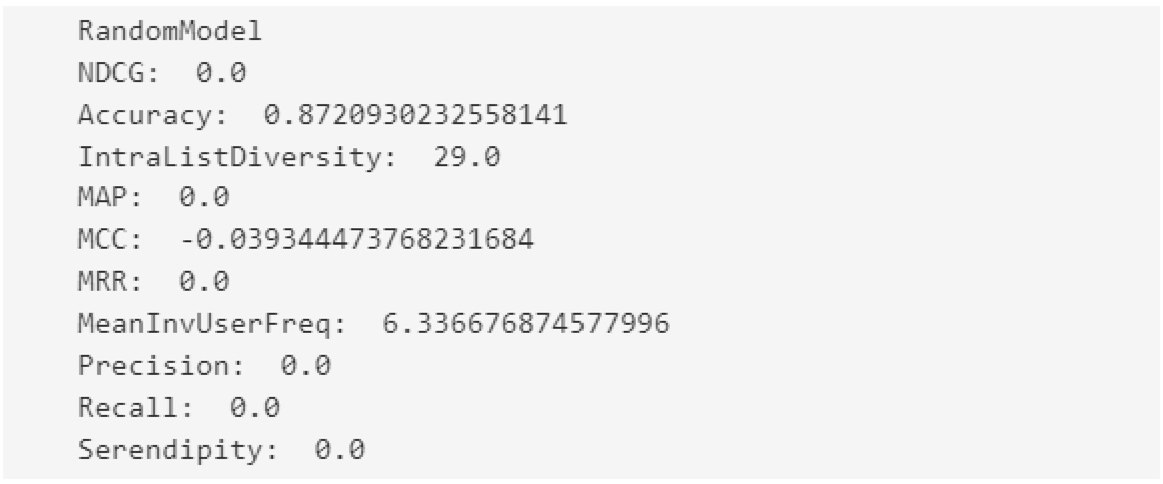
(Additional) RecTools models

On the evaluation stage we compute the following metrics:

1. NDCG (Normalized Discounted Cumulative Gain): Measures the effectiveness of ranked recommendations. A higher NDCG score indicates better-ordered recommendations. Calculated for the top-k recommendations.
2. Accuracy: Represents the proportion of correctly predicted recommendations among the total recommendations. Accuracy is computed for the top-k recommendations.
3. IntraListDiversity: Quantifies the diversity of items within a recommended list. A higher score indicates a more diverse set of recommendations. Evaluated for the top-k recommendations.
4. MAP (Mean Average Precision): Computes the average precision across different levels of recall. MAP is calculated for the top-k recommendations.
5. MCC (Matthews Correlation Coefficient): Measures the correlation between actual and predicted recommendations. The MCC score is computed for the top-k recommendations.

6. MRR (Mean Reciprocal Rank): Calculates the average of the reciprocal ranks of the first correct recommendation. MRR is evaluated based on the top-k recommendations.
7. MeanInvUserFreq (Mean Inverse User Frequency): Measures the diversity of recommended items across users. A lower score indicates higher diversity. Computed for the top-k recommendations.
8. Precision: Reflects the accuracy of positive recommendations among the total recommended items. Precision is calculated for the top-k recommendations.
9. Recall: Measures the ability to retrieve relevant items among all relevant items. Recall is computed for the top-k recommendations.
10. Serendipity: Assesses the novelty or unexpectedness of recommendations. Higher serendipity scores indicate more surprising recommendations. Evaluated for the top-k recommendations.

By consistently choosing $k=10$ for all metrics, the evaluation process ensures a standardized and meaningful comparison of recommendation performance across different models.
Unfortunately, we have no time to solve issues with datasets that is why most metrics are not representative at all. Example of metrics computed for RandomModel:



```
RandomModel
NDCG:  0.0
Accuracy:  0.8720930232558141
IntraListDiversity:  29.0
MAP:  0.0
MCC:  -0.039344473768231684
MRR:  0.0
MeanInvUserFreq:  6.336676874577996
Precision:  0.0
Recall:  0.0
Serendipity:  0.0
```

Main Results

Data

We choose the **`diaries_quotes_emb_reddit_interactions.csv`** dataset consisting of 544 samples (`diary, quote, interaction`) that was created by `gpt-3.5-turbo`.

Model

The best model according to our metrics in the previous section is fine-tuned `cardiffnlp/twitter-roberta-base-emotion-multilabel-latest`. This model obtained **0.485** of `Custom-GPT-Score` , which is by **21.25%** larger than its corresponding baseline (**0.4**).

Difficulties

The diary and quote domain is very difficult to analyze by ML because there is a lack of the datasets to train models. GPT is widely used nowadays [6] for dataset collection. In our case it was challenging to simulate the real user's data, since it is nondeterministic and our domain still looks hard for LLMs. Therefore, we need human evaluation to be conducted for making proper interactions dataset; and for the final models evaluation.

Conclusion

In this project, we developed a recommendation system that leverages NLP and DL techniques to suggest inspirational quotes in response to users' descriptions of their day (we called them diaries). The core methodology involves creating a content-based recommendation framework, where users (diaries) and items (quotes) are matched based on emotional congruence.
The system successfully demonstrates the potential of using NLP and DL techniques in developing a recommendation system tailored to emotional contexts. This project underscores the feasibility and utility of emotion-aware recommendation systems in personalizing user experiences and stands as a promising step towards integrating emotional intelligence into recommendation systems, offering a unique approach to enhancing user experience through personalized and mood-lifting content.

Timeline

1. Data, methods, and tools exploration
2. Problem setting
3. Data preprocessing
4. Models training
5. Models evaluation
6. Creating a demo with a best model

References

[1] J. Camacho-Collados et al., ‘TweetNLP: Cutting-Edge Natural Language Processing for Social Media’. arXiv, Oct. 25, 2022. doi: 10.48550/arXiv.2206.14774. Available: http://arxiv.org/abs/2206.14774. [Accessed: Oct. 9, 2023]
[2] D. Demszky, D. Movshovitz-Attias, J. Ko, A. Cowen, G. Nemade, and S. Ravi, ‘GoEmotions: A Dataset of Fine-Grained Emotions’. arXiv, Jun. 02, 2020. Available: http://arxiv.org/abs/2005.00547. [Accessed: Nov. 6, 2023]
[3] ‘RecTools’. MTS, Nov. 13, 2023. Available: https://github.com/MobileTeleSystems/RecTools. [Accessed: Nov. 10, 2023]
[4] ‘RecTools – OpenSource библиотека для рекомендательных систем’, Хабр, Nov. 10, 2023. Available: https://habr.com/ru/articles/773126/. [Accessed: Nov. 10, 2023]
[5] “MTEB Leaderboard - a Hugging Face Space by mteb.” Available: https://huggingface.co/spaces/mteb/leaderboard. [Accessed: Nov. 28, 2023]
[6] Y. Wang et al., “Self-Instruct: Aligning Language Models with Self-Generated Instructions.” arXiv, May 25, 2023. doi: 10.48550/arXiv.2212.10560. Available: http://arxiv.org/abs/2212.10560. [Accessed: Nov. 28, 2023]