Semantic Song Search

Finding Meaning in Lyrics

Written Report: Shea Conaway
Project Team Including: Mahmoud Hamza, Henry Savich

Data Science Institute, Vanderbilt University

Submitted 13 December 2022
https://github.com/santarabantoosoo/semantic_song_search

Introduction

People have enjoyed music to the beginnings of recorded history, and even prior there is archaeological evidence of musical instruments. Music is important to people as a conveyor of thought and emotion.

In quite recent history, the listener had limited access to songs. Before recordings it was only music their community could produce live, such as church hymns. With radio, access was expanded dramatically but determined by the broadcaster. With vinyl, tapes, and CDs, much was available but still constrained by the budget of the listener.

Now, with a modest monthly subscription, the modern listener can call up virtually any song that has ever been made. The problem has gone from one of scarcity to one of abundance, sifting through an endlessly expanding catalog to find the songs most enjoyable and meaningful to the listener.

We have taken a novel approach to solve this problem. By applying the latest natural language processing technology, we allow the user to search for songs based on the inherent meaning in the song's lyrics. This approach goes beyond basic keyword matching to extract semantics independent of specific wording, connecting songs by topics described by the user in arbitrary detail.

After collecting an extensive lyrics dataset, we apply a pre-trained sentence transformer model to create embeddings for each song. These vector representations of natural language are compared in order to find songs similar in meaning to a user-supplied query. We go further and develop a novel training method to fine-tune the model to achieve better results in the song lyrics domain. Lastly, we consider additional improvements.

Data

We obtained lyrics from the "5 Million Song Lyrics Dataset" on Kaggle.com that had been scraped from Genius.com, a song lyrics and annotations website. The source data was 9.21Gb with 5.9 million rows. In addition to song lyrics, the dataset includes title, artist, year, genre, # views, and id.

The data was cleaned and prepared prior to analysis. Songs that didn't contain lyrics were dropped. We removed songs that contained any non-English characters. This unsophisticated method did leave some non-English songs, but we judged it acceptable. To make the dataset more manageable and the search results more relevant, we limited the full dataset to popular songs with more than 1,000,000 views on Genius.com.

Methods

Computing Resources

We used a Jupyter notebook environment on Google Colab with a paid Pro account for the main computing tasks. This allowed us to utilize a high-RAM virtual machine with a premium GPU. Code, data, and models were stored on Google Drive.

Framework

The theoretical framework of our approach is Sentence-BERT (SBERT), which builds on the BERT model to efficiently extract meaning from entire sentences. It embeds them as vectors that can be compared with cosine-similarity.

SBERT is a transformer model, a form of deep learning that uses self-attention to weight input by its significance, thus focusing on important portions of the data. Transformer models are the state-of-the-art in natural language processing tasks. These concepts are implemented in SentenceTransformers, a Python package based on PyTorch and Transformers.

Base Model

We used the pre-trained sentence transformer model all-MiniLM-L12-v2 from the HuggingFace Model Hub as our base model. We chose it for its balance between performance, speed, and size. It was trained for sentence similarity on a one-billion sentence pair dataset, derived from public sources like WikiAnswers and Stack Exchange.

Fine-Tuning

We fine-tuned the base model to the song lyrics domain. A limitation of the base model all-MiniLM-L12-v2 is that it was trained on mostly literal and technical texts. In contrast, song lyrics can be indirect, metaphorical, and poetically short.

Without the resources to construct an additional dataset of semantic pairs, we devised a novel method for a training set that avoided supplying our own preconceived notions of meaning, allowing the algorithm to find similarities in songs with minimal supervision.

We created training pairs from song halves in our Genius.com dataset. Songs should have a cohesive meaning and a running narrative from start to end. Being able to match song halves should correlate with finding similar meaning in different wordings. To avoid simplistic matching, we removed repeated verses for more distinct song halves.

The loss function we used for fine-tuning training was MultipleNegativesRankingLoss. This loss accepts batches of one anchor, a positive pair match, and multiple negatives. It then minimizes the negative log-likelihood for softmax normalized scores. In essence this penalizes modeling steps that increase the probability of matching the anchor to a negative pairing instead of the true positive.

We trained two models, genius-MiniLM-128 and genius-MiniLM-510, one with the default 128 token limiter and another with a maximally expanded 510 token limiter. This varies the portion of the songs the model is able to process.

Embedding Procedure

To illustrate the concept of embedding, we provide a toy example of a two-dimensional semantic vector space in Figure 1, where words can be compared only by the two dimensions of entertainment and importance. Here, "youtube" and "textbook" are semantically different and almost diametrically opposed. The words "lecture" and "textbook" are much more similar with a smaller angular difference.

In this project, songs were embedded into 384-dimensional semantic vector spaces, using the base and two fine-tuned models. These dimensions are derived during the training process to be highly dense in meaning and, unlike the toy example, are not human-interpretable.

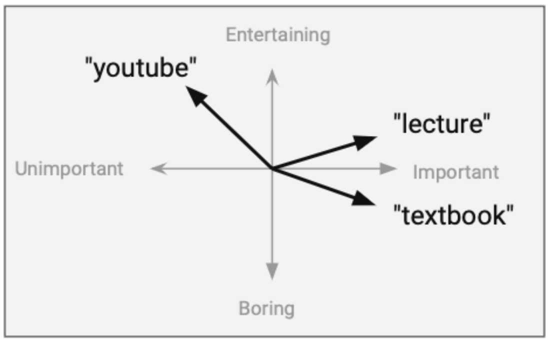


Figure 1: A toy 2-d semantic vector space

Comparison and Search

Once the songs have been embedded as vectors in the semantic space, they can be compared simply using cosine similarly. Language with similar meaning has a similar vector representation. Therefore, the angle between the vectors is small. The cosine of this angle ranges from 1 to -1, where 1 is no difference (semantically equal) and -1 is a vector pointed in the opposite direction (semantically opposed).

To search for a song by topic, we simply embed one more text - the user query - and then calculate its cosine similarity to each of the existing song embeddings. Cosine similarities can be computed very quickly for existing embeddings. The results are presented by ranking the top similarity scores

Assessment

We assessed two measures of performance for comparison of the models. One was an objective measure of accuracy by matching song halves on a validation set. The second was our subjective assessment of queries and their resulting song lyrics.

Results

In Table 1, we provide the results of our objective measure, the summary accuracy scores from our validation set comparison. In Figure 2, we provide one example of the types of queries used in our subjective assessment.

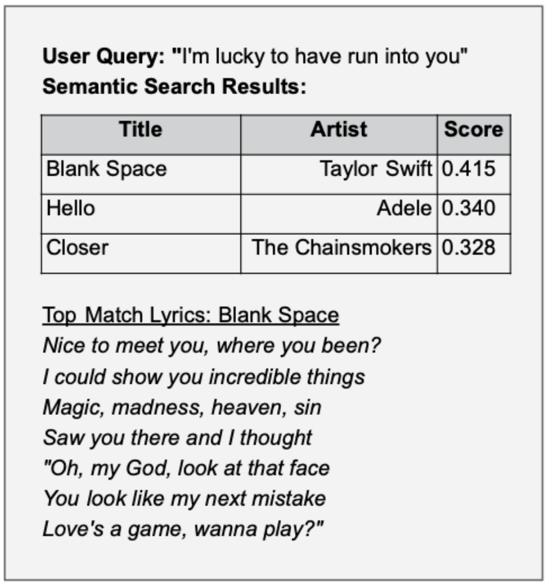
Base Model

all-MiniLM-L12-v2 showed high objective and subjective results. It matched song halves with 0.951 accuracy. In addition, the subjective song lyrics results appeared to us highly related to a variety of queries.

Table 1: Comparison of models

Model	S2 Accuracy	Subjective Results
all-MiniLM-L12-v2	0.951	Good
genius-MiniLM-128	0.990	Good / Uncertain
genius-MiniLM-510	0.998	NA

As a single example, Figure 2 shows some of the impressive possibilities of semantic song search. The top search result captures the main concept of the user query (an exciting new relationship), but also subtle allusions to "luck" (magic and games).



User Query: "I'm lucky to have run into you"
Semantic Search Results:

Title	Artist	Score
Blank Space	Taylor Swift	0.415
Hello	Adele	0.340
Closer	The Chainsmokers	0.328

Top Match Lyrics: Blank Space
Nice to meet you, where you been?
I could show you incredible things
Magic, madness, heaven, sin
Saw you there and I thought
"Oh, my God, look at that face
You look like my next mistake
Love's a game, wanna play?"

Figure 2: Example query results from base model

Fine-Tuned Models

Our fine-tuned models, genius-MiniLM-128 and 510 increased performance on the song halves measure to near perfect. However, it was difficult to detect improvement in subjective results for genius-MiniLM-128, and there were a few spurious outputs. For genius-MiniLM-510, the marginal objective accuracy bump did not justify its computational demands, so it was not implemented fully for subjective assessment.

Discussion

The results show that the base model performs well, such that improving on it measurably will take a more considered fine-tuning and validation methodology. Our initial attempt genius-MiniLM-128 achieves greater objective results, but it was difficult to find meaningful improvement in the subjective results.

For fine-tuning within the song-parts paradigm, possibilities for better training include more scrubbing of similar language between song halves, since removing duplicate verses still leaves similar wording. Different song-parts sizes could be explored for pairing, from quarters down to individual verses.

Moving beyond endogenous training pairs, collecting other texts for pairing with the song lyrics, like annotations from SongMeanings.com or YouTube comments on music videos, could better capture the needs of our use case. User queries tend to be short, which are then matched to longer lyrics. Training data that better approximates this usage could bear greater results.

Conclusion

In short, using public open-source tools and readily available data, it's possible to implement semantic song search with relative ease. Using a pre-trained model alone achieves viable results. Fine-tuning a model using our novel training approach produced positive objective accuracy but requires additional work to return unambiguously superior query results. Exploring other approaches is also warranted.

Personal Contributions

Our team worked together well, and each member made substantial contributions. I am very grateful for the work of Mahmoud Hamza and Henry Savich on this project.

Logistically, I arranged our team meetings, kept our notes and agendas, and created our shared project environment. Once we had decided on the semantic song search topic, I found our dataset and proposed our fine-tuning methodology. I coded the fine-tuning and validation notebooks. Across our series of presentations, I discussed data, project goals, and methodology.

References

- Kostas Stathoulopoulos. *How to Build a Semantic Search Engine With Transformers and Faiss*. Nov 2020.
- Nikhil Nayak. *5 Million Song Lyrics Dataset*. Apr 2022.
- Lewis Tunstall, Leandro von Werra, Thomas Wolf. *Natural Language Processing with Transformers*. June 2022.
- Nils Reimers, Iryna Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT Networks*. Aug 2019.
- Nils Reimers. *SentenceTransformers Documentation*. Dec 2022.
- *all-MiniLM-L12-v2*.
- Omar Espejel. *Train and Fine-Tune Sentence Transformers Models*. Aug 2022.
- James Briggs. *Natural Language Processing (NLP) for Semantic Search*. Dec 2022.
- Nils Reimers. *Training State-of-the-Art Sentence Embedding Models*. Jun 2021.
- Saketh Kotamraju. *An Intuitive Explanation of Sentence-BERT*. Jun 2022.