Clustering Spotify Podcasts with NLP-Driven Insights

Data Collection, Cleaning, and Tokenization:
Using Selenium and Spotify API, we scraped the top 50 podcasts per genre (`fetch_top_podcast.py`). Metadata was retrieved and filtered for English podcasts (`fetch_podcast_details.py`), followed by episode details (`fetch_episode_details.py`), yielding 284,481 episodes. Episode descriptions were cleaned and tokenized (`clean_description.py`), performing normalization, URL removal, lemmatization, and stopword removal. Tokens for each podcast were consolidated into "frequency" vectors relative to a global vocabulary of 47,718 tokens (total number of unique tokens for all podcasts).

Related Tokens	Frequency	Unrelated Tokens	Frequency
murder	47	Technology	1
crime	33	Sleep	0
killers	21	Comedy	0
cover	12	Finance	0
mysterious	5	Cooking	0
survival	3	Science	1

Table 1: Frequency vector for a true crime podcast, showing high frequencies for related tokens and very low or zero frequencies for unrelated tokens.

Computing metrics:
Three metrics were computed (`compute_metrics.py`) using frequency vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ (both of length 47,718) for any two podcasts as follows:

1. *Normalized Total Feature Similarity:* Measures cosine similarity between two frequency vectors.

$$\mathrm{NTFS}(\boldsymbol{x},\boldsymbol{y}) = \frac{\langle \boldsymbol{x},\boldsymbol{y}\rangle}{||\boldsymbol{x}||_2\ ||\boldsymbol{y}||_2} \in \mathbb{R}_{[0,1]} \quad \longrightarrow \quad \text{higher implies more directional similarity}$$

   Strengths: Robust for sparse vectors. Weakness: Assumes all tokens equally important.

2. *Jaccard Token Similarity:* Compute metric signifying proportion of overlapping tokens.

$$\mathrm{JTS}(\boldsymbol{x},\boldsymbol{y}) = \frac{\sum \min(x_i, y_i)}{\sum \max(x_i, y_i)} \in \mathbb{R}_{[0,1]} \quad \longrightarrow \quad \text{higher implies more token overlap}$$

   Strengths: Simple and interpretable measure of overlap. Weakness: Sensitive to scaling.

3. *Weighted Token Diversity Similarity:* Uses L1-normalized frequency vectors that emphasizing token diversity.

$$\mathrm{WTDS}(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{n} \sqrt{\frac{x_i}{||\boldsymbol{x}||_1} \cdot \frac{y_i}{||\boldsymbol{y}||_1}} \in \mathbb{R}_{[0,1]} \quad \longrightarrow \quad \text{higher implies more shared diversity}$$

   Strength: Highlights diversity. Weakness: Assumes uniform importance across tokens.

Recommendation algorithm:
Given a selected podcast k, generate n-recommendations from a list of T podcasts as follows (scatter-plot.py). We construct a vector of 3-dimensional tuples of similarity metrics, for all $i = 1, \ldots, T$.

$$\begin{array}{cc} & \begin{array}{ccccc} \text{podcast 1} & \ldots & \text{podcast k} & \ldots & \text{podcast T} \end{array} \\ \text{podcast k} & \left(\begin{array}{ccccc} \mathscr{S}_{k,1} & \ldots & \mathscr{S}_{k,k} & \ldots & \mathscr{S}_{k,T} \end{array} \right) \end{array}$$

$$\text{where, } \mathscr{S}_{k,i} = \begin{cases} \big(\ \mathrm{NTFS}(\boldsymbol{x_k},\boldsymbol{x_i}),\ \mathrm{JTS}(\boldsymbol{x_k},\boldsymbol{x_i}),\ \mathrm{WTDS}(\boldsymbol{x_k},\boldsymbol{x_i})\ \big) & \text{if } i \neq k \\ (1,1,1) & \text{if } i = k \end{cases}$$

Next, we quantify dissimilarity by computing the euclidean 2-norm distance with respect to podcast k:

$$d_{ki} = ||\underbrace{(1,1,1)}_{\mathscr{S}_{k,k}} - \mathscr{S}_{k,i}||_2 = \sqrt{\big(1-\mathrm{NTFS}(\boldsymbol{x_k},\boldsymbol{x_i})\big)^2 + \big(1-\mathrm{JTS}(\boldsymbol{x_k},\boldsymbol{x_i})\big)^2 + \big(1-\mathrm{WTDS}(\boldsymbol{x_k},\boldsymbol{x_i})\big)^2}$$

Finally, we sort by distance (lowest to highest) and report the n-closest podcasts. Each reported podcast represents those whose description match most closely in direction, shared content coverage, and diversity of content to podcast k, ensuring tailored recommendations for enhancing user engagement.