The Alan Turing Institute

Data Study Group Final Report: Entale

19 – 30 Apr 2021

Recommendation systems for podcast discovery

https://doi.org/10.5281/zenodo.5818331

Contents

1 Executive summary

1.1 Challenge overview

Entale is a small London-based startup bringing together product designers, engineers, data scientists, producers, and journalists whose goal is to transform the podcast listening experience through the Entale app. Entale is uniquely positioned to tackle podcast discovery, having developed robust pipelines for collecting, storing, and querying data from hundreds of thousands of podcast RSS feeds. In addition, Entale has released an in-house information retrieval system that extracts validated contextual content from podcast transcriptions in the form of linked named entities and external links added by the podcaster. Over time, Entale has also accrued a considerable volume of in-app user listening data.

The Entale challenge was composed of two interrelated goals. Firstly, we aimed to develop methods for capturing relationships between podcasts, incorporating information about the content discussed within them. Secondly, we worked to produce podcast recommendations, some of which utilised these inferred relationships – one key desired aspect of these recommendations was that they should allow 'rabbit hole' discovery experiences, where users could be pushed to further explore topics of interest.

1.2 Data overview

The data was divided into two types, user data and podcast data. Unique show, episode and show identifiers helped track information across multiple datasets. A set of episode descriptions and transcriptions were pre-processed to extract named entities. While each dataset was large, the overlap between episode listens, show subscriptions and episode transcriptions was less than 10%, making it challenging to fully incorporate user data with detailed episode content.

1.3 Main objectives

The main objective of the Entale challenge is to create a podcast recommendation system that gives listeners a “rabbit hole” journey from podcast to podcast. Rather than use the recommendation system to reinforce a listener’s preferences, our objective is to recommend new podcasts that are a comfortable stretch for the listener. Our research aims for the Data Study Group are:

- To use listener history to understand users’ intentions and preferences.
- To experiment with different approaches to topic modelling.
- To develop a modular pipeline that combines a user’s profile with episode features in multiple ways, providing Entale with several recommendation approaches to test with users.
- To identify podcast and listener features that may be useful for future work.

1.4 Approach

A broad overview of our approach is available in Figure 1. In summary, the provided data had some issues with fully integrating the item datasets and the user datasets due to lack of overlap. As such, to begin with we focused on separately working with user data (leading to methods such as collaborative filtering, and general user embeddings) and episode data (using *e.g.* topic models, word embeddings *etc.*) - we later partially recombined these methods, and evaluated different options against each other where possible.

We have developed a modular pipeline (see Figure 1) that incorporates a range of recommendation algorithms. We distinguish broadly between content-based and user-based recommendations, though each pipeline can combine listener and episode data. Incorporating descriptions, transcripts, and entities covered in episodes, we implement topic models and network-based approaches to recommend episodes that are similar to a user’s last episode or their entire listening history. Complementing these approaches, we include collaborative filtering which takes into

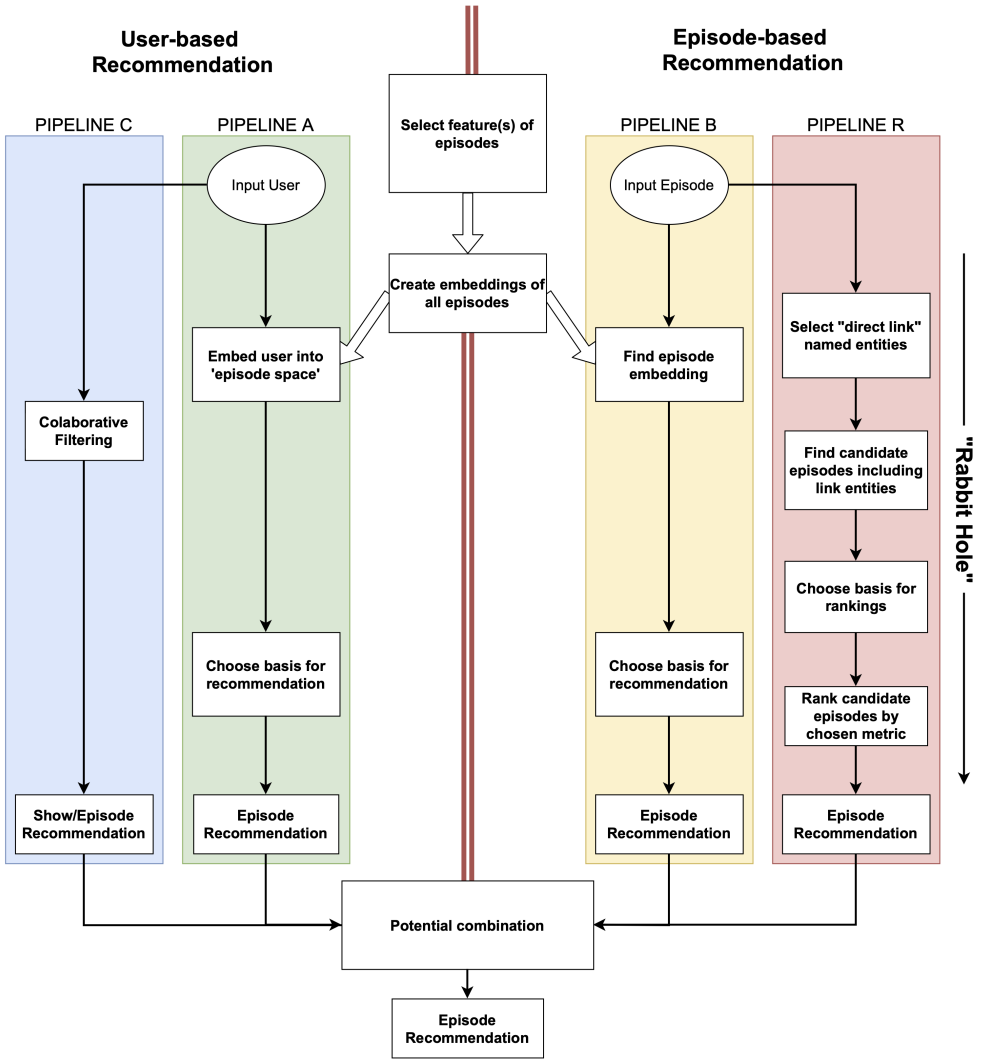


Figure 1: Overview of the recommendations pipelines under consideration

account the listening behaviour of all users as a whole. Our approach recognizes that many different aspects matter for podcast recommendation, ranging from the podcast's style and tone, to show popularity and a user's willingness to discover diverse topics. For example, we develop a special rabbit hole system which takes users beyond their current realm of interests and which may be most exciting for brave listeners. The modular approach allows for pieces of the pipeline to be activated or deactivated for different users and at different stages of the product cycle. Finally, our models are ready for evaluation with real users, whose preferences will ultimately determine the components of the recommender pipeline.

1.5 Main results

First and foremost, our work has highlighted important aspects of the provided datasets, and how they can be interrelated. For instance,

- Linking episode transcription data – and derived features such as extracted named entities – to user listens would allow detailed insights into typical user profiles, and help provide personalised recommendations.
- Podcast categories correspond only weakly to the content within them – in particular, for a variety of podcast embedding methods, we observed local clustering of categories within topics, but no global clustering. In other words, podcasts within the same category might discuss a certain theme in a similar way, but other categories could discuss the same topic. For instance, podcasts within both Society & Culture, and Business categories may both discuss recent developments in AI, but focus on different aspects.
- Additional features, such as those from direct analysis of the audio (pitch, volume, number of speakers) and sentiment analysis could be useful, but benefits ought to be carefully weighed against computational cost and possible bias.
- Much more simple features, such as whether a user has either

saved/'liked' a podcast, or indeed actively said that they do not wish to be recommended a podcast/see it on their home screen, could be at least as useful signals, with much lower cost associated to their collection. Some of this information, for instance particular shows and categories they're interested in, could be collected during an 'onboarding' process prior to first use of the app.

Overall, our pipeline follows that typical for recommender systems: candidate generation, scoring, and (thus far to a lesser extent) re-ranking. We also found that

- Relatively simple NLP approaches on named entities still allow for reasonable 'rabbit hole' recommendations, through combination with simple constraints – in particular that the recommended podcasts contain references to a certain entity of interest.
- Quantitative evaluation of recommendations is an inherently difficult task – while several methods have been proposed herein, as discussed in Section 6.2, we emphasise that the ideal approach should be A/B testing with Entale users. Especially as at the time of writing the app has no recommendation feature, and we would expect user behaviour to alter in the presence of such.

1.6 Limitations

As the Entale user base changes, the overall pipeline as well as the selected models may need to be adapted to account for new types of user behaviour, new niche or mainstream interests, or other factors related to user preferences.

There was found to be a lack of overlap across some of the datasets, limiting the kinds of recommendation approaches that were feasible. We therefore recommend future work to focus on exploring some of the dataset combinations that were not addressed in this report.

At the time of writing there is no recommendation system in place, meaning the possibilities for evaluation are limited. The most rigorous approach - A/B testing with Entale users - was not feasible given the time constraints. We have addressed this challenge, by considering multiple evaluation approaches. For example, we assess how well our system reproduces

true listens, with the caveat that users currently receive recommendations through social networks, charts, or active discovery, rather than a baseline recommendation system.

In general, almost all methods applied include *hyperparameters*, *i.e.* parameters chosen prior to application that affect the output of the method. In many cases, these can have a substantial impact on the quality of the output, and so careful tuning is vitally important. Unfortunately, given the time constraints of the project there was little opportunity to conduct such tests, and so there are some issues in directly comparing the methods applied to one another to choose the 'best' – we leave this to future work.

1.7 Recommendations and future work

Our recommendations for future work can be grouped into three categories: improvements of our models, evaluation, and development of new approaches. They are briefly listed below and described in more detail in section 8.

Model improvements include:

- Hyper-parameter tuning for all methods.
- Incorporate information on charts / popularity, at the least in ranking of recommendations (and possibly as user segmentation feature, *e.g.* whether user is more niche or mainstream).
- Expanded data collection, including more nuanced and granular user data.

Evaluation of these and other models may cover:

- A/B testing of different types of recommendation, *i.e.* by category, topic, changing homescreen, through entity exploration etc.
- Qualitative assessment of benefits of each model, and methods to recombine into singular lists of recommendation (*e.g.* user-specified proportion of top-k recommendations for each) – super learner methods?

Finally, we collected a range of other models that may help improve recommendations. These include:

- Relatedly, re-ranking methods for proposed recommendations to ensure *e.g.* diversity.
- Recurrent neural networks for time series forecasting, given longer time series of user data.
- Additional feature extraction, for example relating to style and tone of podcasts, sentiment, or speaker diarisation.
- Reinforcement learning to help improve recommendations for individuals.

2 Data overview

The dataset is organized into two major sets, podcast items and podcast users.

2.1 Items

The item set contains a sample of approximately 32,000 episodes of English-language podcasts from the iTunes database, content extracted from these podcasts, and associated metadata. The episodes are restricted to those published between January 2016 and early February 2021.

Episodes lie at the center of a three-level hierarchy: shows, episodes, and entities/URLs. A podcast show is serialized into one or more episodes. An episode is associated with one or more named entities and URLs. Named entities are machine-extracted from episode descriptions and transcriptions, whereas URLs are machine-extracted from episode descriptions only.

The item data is organized into five CSV files: shows, episodes, transcription entities, description entities, description URLs. Each of these subsets is described in more detail below.

Episodes and shows are associated with unique identifiers. Since at least one of these identifiers is present in every subset, they can be used to match records across CSV files.

2.2 Podcast Items

2.2.1 Episodes

Episodes were sampled semi-randomly to counterbalance their representation across primary thematic iTunes categories (see Appendix A for a complete list of categories).

The sampling algorithm did not limit the number of episodes that could be sampled per show, so episodes from shows that released more episodes during the examined time period were more likely to be sampled.

A small percentage of episode descriptions (just over 1%) were in languages other than English. These were detected using the langdetect python module, then filtered out prior to descriptions being used for modelling.

variable	type	description
`episode_id`	`str`	alphanumeric code (digits, lowercase & uppercase characters) uniquely identifying an episode across item & user sets
`show_id`	`str`	numeric code uniquely identifying a show across item & user sets
`release_date`	`str`	date on which episode was released by publisher, formatted as `yyyy-mm-dd`.
`episode_title`	`str`	title of episode given by publisher, no post-processing
`episode_description`	`str`	teaser/summary of episode content provided by publisher, sometimes includes follow-up links & advertisements, post-processed to remove HTML tags, string trailing spaces and newlines, normalize UTF-8 encoding to NFKD
`transcription`	`str`	transcription of episode audio files

2.2.2 Shows

The shows in this set are those which have at least one episode in the episodes set (section 2.2.1). There are 1670 unique shows in the set, with approximately 19.1 episodes per show on average.

For a complete list of show categories, see Appendix A.

The ranks are specific to a category. For a given show, the ranks are always associated with the same category across UK and US charts. Given that the shows are a random sample of those in the iTunes database, only some charting shows are included; the complete ranking is thus not represented in this set.

variable	type	description
show_id	str	numeric code uniquely identifying a show across item & user sets
category	str	primary thematic iTunes category (see Appendix A)
rank_uk	int	category rank in UK iTunes char, no cutoff
rank_us	int	category rank in US iTunes chart, no cutoff
show_title	str	title of show given by publisher, no post-processing
show_description	str	general summary of show/show teaser provided by publisher, post-processed to remove HTML tags, string trailing spaces and newlines, normalize UTF-8 encoding to NFKD

2.2.3 Transcription Entities

Transcription entities are named entities machine-extracted from transcriptions included in the episodes set and linked to real-word entities

via the web. Some entities that appeared in transcriptions were excluded if they are included in an entity blacklist. The accuracy of tags and links, automatically induced, is not perfect, but, overall, sufficiently robust for modeling. This is especially apparent in the case of web links for named entity spans that required disambiguation (e.g. Brian Cox, the physicist vs. Brian Cox, the actor).

variable	type	description
episode_id	str	alphanumeric code (digits, lowercase & uppercase characters) uniquely identifying an episode across item & user sets
span	str	extracted text representing named entity
label	str	named entity label assigned by NER model, see Appendix B.1 for a list of possible entity labels
web instance	str	Web P31 type property, class of which this subject is an example & member
web url	str	link to website about named entity
web extract	str	top paragraph of website about named entity

2.2.4 Description Entities

Description entities are named entities machine-extracted from episode descriptions included in the episodes set. Description entities were identified and tagged using a different Named Entity Recognition model from the ones used for transcription entities, so there may be differences

not only in labels, but also what falls under the scope of a given label. See Appendix B.2 for a list of possible entity labels.

As for transcription entities, some entities appearing in descriptions were excluded if they are included in an entity blacklist.

variable	type	description
episode_id	str	alphanumeric code (digits, lowercase & uppercase characters) uniquely identifying an episode across item & user sets
span	str	extracted text representing named entity
label	str	named entity label assigned by NER model, see Appendix B.2 for a list of possible entity labels
web instance	str	Web data P31 type property, class of which this subject is an example & member
web url	str	link to website about named entity
web extract	str	top (summary) paragraph of webpage about named entity

2.2.5 Description URLs

Description URLs are HTML links extracted from episode descriptions, often containing references to topics being discussed in the episode, follow-up resources such as articles or books, publisher privacy policies or hosting platform and/or third-party advertisements.

If an episode from the episode set does not contain any URLs in the description, it will not be listed in this set. If an episode contains multiple URLs in the description, it will appear as many times as there are URLs in

the descriptions (i.e. one record per URL).

variable	type	description
episode_id	str	alphanumeric code (digits, lowercase & uppercase characters) uniquely identifying an episode across item & user sets
episode_description_url	str	HTML links detected in episode description prior to text cleaning.

2.3 Users

The user set contains two types of anonymous user interactions with podcast content: episodes a given user listened to over a specific period of time (episode listens), and shows they subscribed to (show subscriptions). These are listed in separate files.

Although episode and show IDs are unique across item and user sets, we cannot guarantee a substantial overlap between episodes and shows in item and user sets. Item and user data is thus best modeled independently.

While episode listens are restricted to a specific time period (12 months) and do not include Entale-produced episodes, show subscriptions have no time or podcast content type restrictions and constitute a complete listing of subscription-type interactions.

2.3.1 Episode Listens

Episode listens contain a list of episodes listened to by anonymized users of the Entale app over a period of 12 months. If a user listened to the same episode on different dates, those will be listed as separate records.

The list does not contain episodes from shows released by our in-house

podcast publisher, Entale Studios. There are nearly 9,300 unique users in the set who have listened to a total of over 72,600 episodes, with a user listening to approximately 7.8 non-Entale produced episodes on average.

variable	type	description
user_id	str	alphanumeric user reference anonymized by hashing, uniquely identifying a user across all episodes they listened to or shows they subscribed to
episode_id	str	alphanumeric code (digits, lowercase & uppercase characters) uniquely identifying an episode across item & user sets
show_id	str	numeric code uniquely identifying a show across item & user sets
listen_date	str	date on which an episode was listened to by a particular user, in the format yyyy-mm-dd; if the same episode was listened to on multiple dates, all will be listed. Note that due to an input error, the dd data is not accurate and set to the 1st of each month.
episode_title	str	title of episode given by publisher, no post-processing
episode_description	str	teaser/summary of episode content provided by publisher, sometimes includes follow-up links & advertisements, post-processed to remove HTML tags, string trailing spaces and newlines, normalize UTF-8 encoding to NFKD

2.3.2 Show Subscriptions

Show subscriptions contain a list of shows each anonymized user subscribed to in the Entale app. This feature allows users, for example, to keep track if their favourite show has released a new episode. The subscriptions are not restricted to a time period and are a sample of show subscriptions in the app.

There are over 24,700 unique users in the set who subscribed to nearly 11,000 unique shows.

variable	type	description
user_id	str	alphanumeric user reference anonymized by hashing, uniquely identifying a user across all episodes they listened to or shows they subscribed to
show_id	str	numeric code uniquely identifying a show across item & user sets
category	str	primary thematic iTunes category (see Appendix A)
show_title	str	title of show given by publisher, no post-processing
show_description	str	general summary of show/show teaser provided by publisher, post-processed to remove HTML tags, string trailing spaces and newlines, normalize UTF-8 encoding to NFKD

2.4 Data quality issues

2.4.1 Dataset Overlap and Depth

In some cases, data overlap is limited. Over 70 % of the shows in the items dataset show up at least once in the listens dataset, but this is the case for only 7 % of the episodes. Of the 9,282 unique users in the *episode_listens* dataset, 62 % (5,746) also appear among the 24,748 users in *show_subscriptions*. Only 9% of episodes in the listener history have transcription entities and transcriptions.

2.4.2 Generalisability from User Data

Entale users are unlikely to be representative of the wider podcast listeners. As a new application only available for iOs, Entale is more likely to attract early adopters and is only accessible to users of Apple devices. The novelty of the platform itself may also impact how users engage with the podcasts as they learn to navigate the system.

One of Entale's strategies for attracting new users is to produce celebrity-based podcasts. Users engaging with the platform may be more interested in that genre of podcast than the general public.

3 Data Visualisation

3.1 Category Analysis

First we filter out data points that we don't have a label for their category. While we have 19 categories in total, Figure 5 shows that even users with more than 100 listens have listened to less than 8 categories on average. However, this may be because some of the categories have really few listens, particularly in the categories of 'Government', 'Leisure' and 'Religion and Spirituality' (Figure 6). The most popular categories for the Entale user base are 'True Crime', 'Society and Culture' and 'Comedy'. Podcasts do often belong in different categories, but the data contained only the category in which a given podcast was ranked highest.

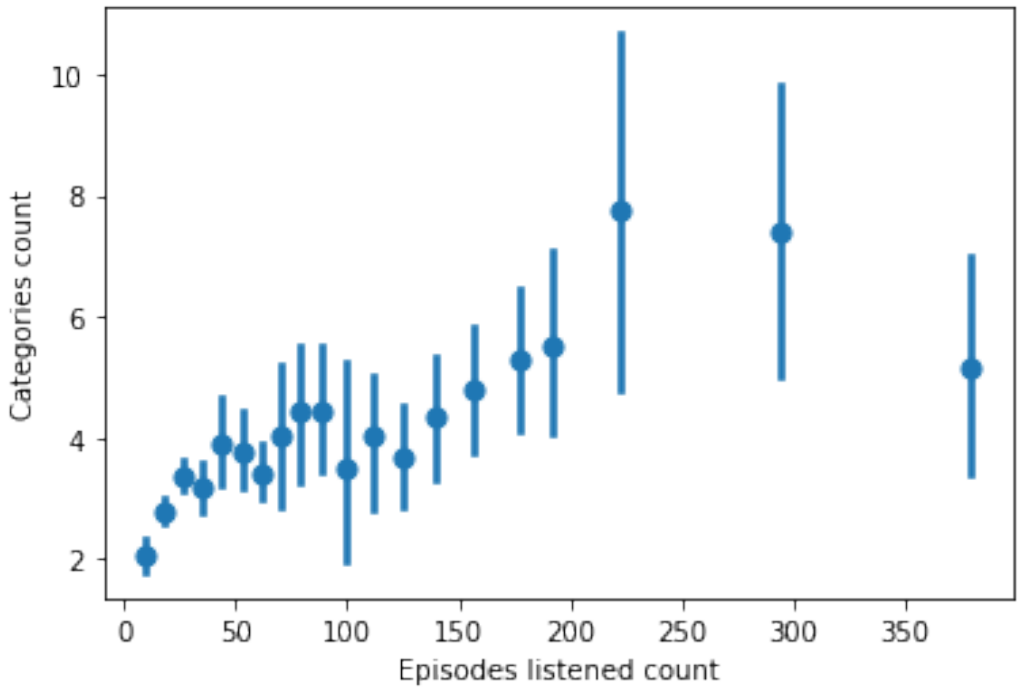


Figure 5: Number of unique categories over number of episodes listened.

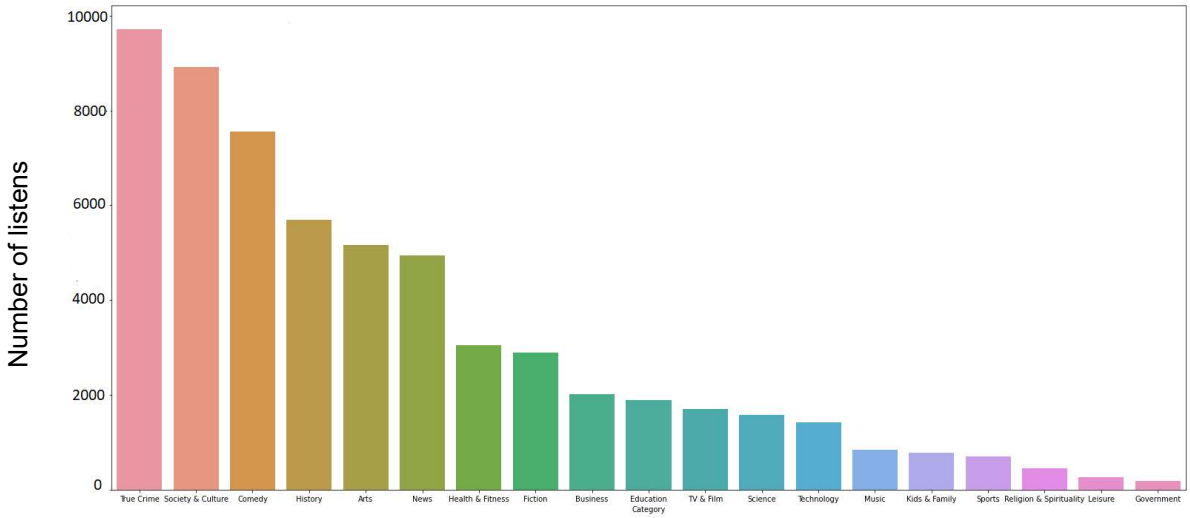


Figure 6: Number of listens for each category.

Number of listens to each of the categories also varies in time. This suggests that listens follow some seasonality patterns (Figure 7). The top-rated category, True Crime peaks in February and climbs to a consistently high number of listens May and October, before appearing to drop off. Comedy also seems to reach peak listenership starting in May, but sees a steep drop off in September. Because the data only covers a year, it's impossible to say whether these patterns occur every year or are due to the release of a very popular show that many listeners follow. One way of disambiguating this in future work would be to compare number of listens per category to the unique number of episodes by category.

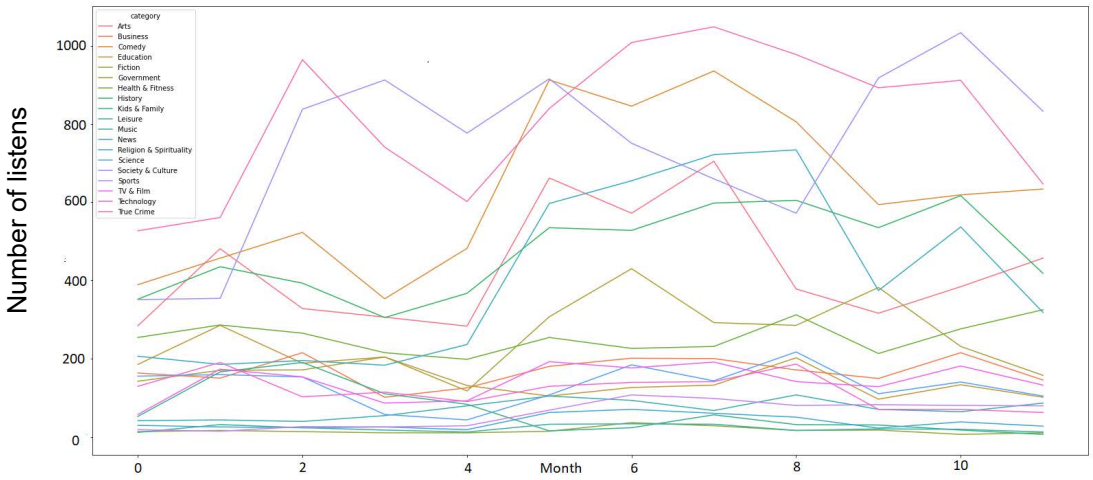


Figure 7: Trends in listens for different categories.

We could also find category similarities by constructing co-listens network of different categories. Figure 8 visualises the graph using a force directing algorithm named Fruchterman-Reingold [1].

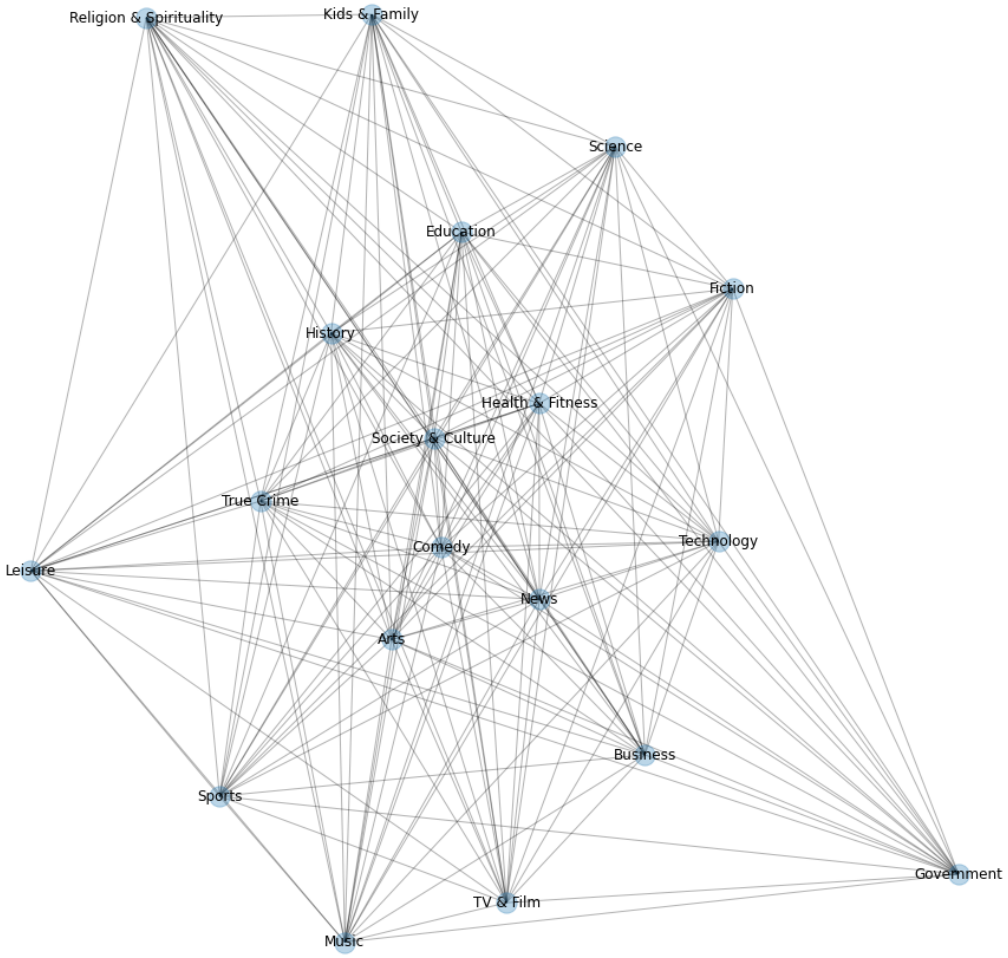
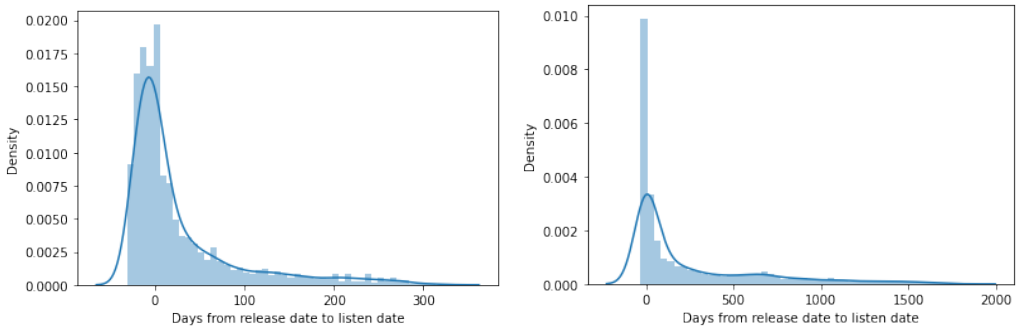


Figure 8: Visualisation of category graph using force directing algorithm.

Another aspect of episodes listens data we examined is what is the relationship between the release date of an episode with its listens. We should consider that we only have listens data for one year period. Bearing that in mind, we first selected the episodes that were published in the same year as our listens. Then we plot the distribution of number of days between the listen event and the episode's release date (Figure 9a). The fact that we have negative numbers up to -30 is because the listening dates are accurate just to month number. We could see a clear pattern

indicating listens mostly happen within a close period to the podcast release date. While because of the time period of listens data, we don't have past listens for previously released episodes, we will still find a similar pattern if we consider all of the episodes (Figure 9b).



(a) Distribution for episodes published in listen data time period

(b) Distribution for all episodes

Figure 9: Distribution of number of days elapsed between episode released and a user listened to it

3.2 Subscriptions

The number of subscriptions for each user is also skewed. While there exist a user that has subscribed to 1191 shows, 55% of users that have subscription data have only subscribed to one show.

If we do the same analysis as of Figure 5 on subscription data we would get Figure 10, which also shows users subscribe to relatively few numbers of categories.

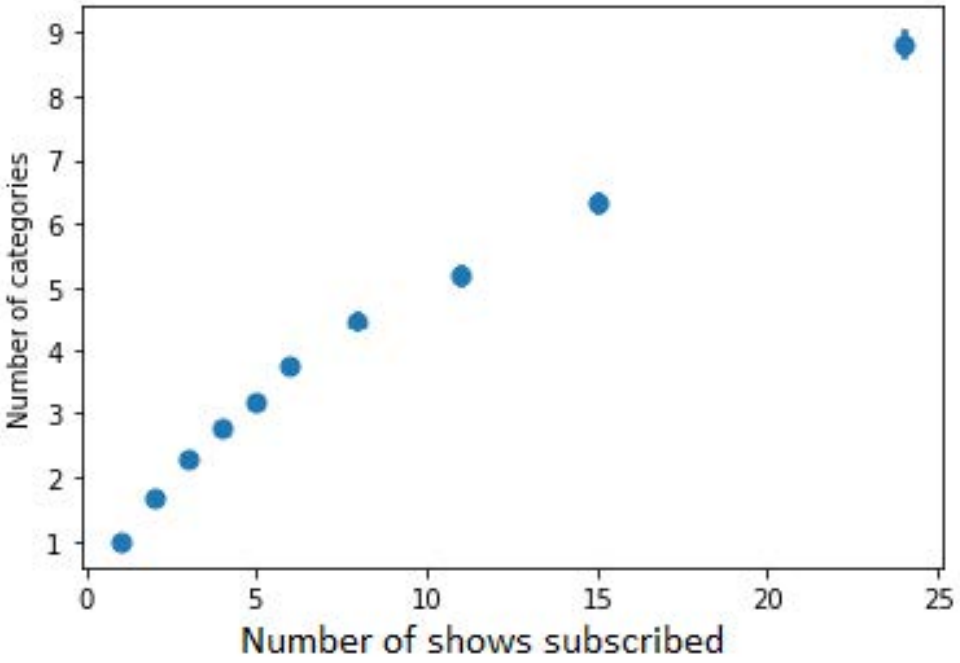
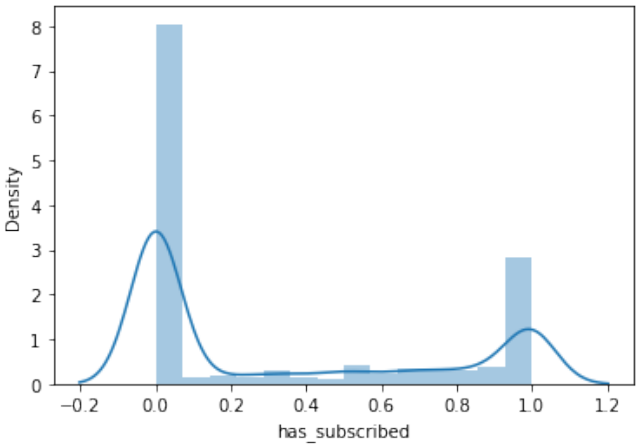
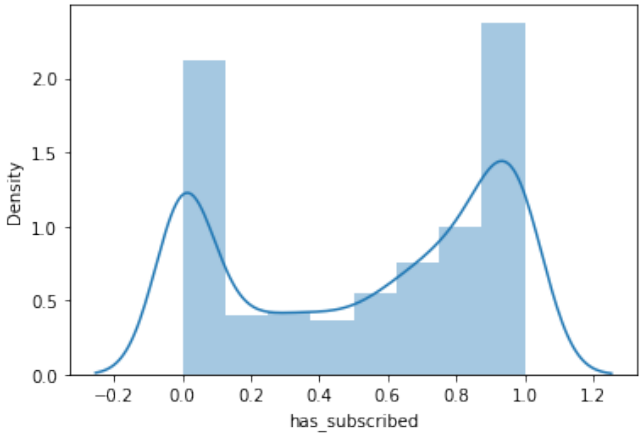


Figure 10: Number of unique categories over number of episodes subscribed.

Another question worthy to answer is how does the users' behaviour vary in terms of how much of their listens are from shows that they have subscribed to. Figure 11 shows the distribution of this ratio for different users both for all users and those that have more than 10 listens.



(a) Distribution for all users



(b) Distribution for users that have more than 10 listens

Figure 11: Distribution of the ratio of listens that the user has subscribed to the show

These distributions of listens, subscriptions, show categories and other aspects of the data carry important implications for podcast recommendation. Considering the variability in user engagement, for

example, the system may need to provide recommendations of different lengths, at different frequencies, and with varying content types to different users. However, as mentioned in section 2.4, the data visualized above is very likely not representative of the current Entale user pool and even less so of the entire population of podcast listeners. Further data collection and analysis are therefore needed to explore some of the implications of the user distributions and to answer questions such as the role of seasonality, the emergent properties of podcasts and user habits, or whether users engage in multiple types of listening behaviour. In the next section (4), we begin diving into some of these questions, by providing hypotheses about user behaviour that have shaped the remainder of our analysis as well as some of the derived datasets developed during the challenge.

4 User Analysis

User behaviour is an important aspect for podcast recommendation. Based on assessment of the available show subscriptions and episode listen datasets, we provide hypotheses on the range of user behaviour that may be encountered during recommendation. We leverage these hypotheses to construct derived datasets as inputs to some of the models discussed in more detail later in this report.

4.1 User Behaviour Hypotheses

Users' patterns of subscriptions and listens can enrich our understanding of how they relate to podcast content. Subscriptions speak to the intent to engage with content in the future, reflecting a spirations a nd wishes. Actual listens show how a user chooses to spend their time. The Entale dataset does not contain information on the duration of a listen, making it challenging to impossible to distinguish whether a listen is cursory or of the full episode. Two or more listens of the same episode do suggest that the user has fully engaged with the episode, whether they are re-listening or returning to a partially completed episode.

Ultimately, recommendations should be for podcasts a user will enjoy. In absence of an existing recommender system and user preference ranking, we rely on incomplete information to predict future user listens. Currently, users see a subset of the universe of episodes, for example when they receive suggestions from friends or when browsing the discovery page, and choose to listen to episodes within this subset. We do not know which recommendations were discarded, but we have show subscriptions, which provide basic insight into a user's realm of podcast visibility. In the sections that follow, we provide hypotheses and observations from qualitative data analysis that aim to give context to using current user behaviour as input to a recommender system. We interpret listening and subscription behaviour theoretically and identify potential data points required to test our hypotheses.

4.1.1 Podcast Format and Cherry-picking

Format, along with duration, may be important features for podcast recommendations given their importance for podcast producers. Podcast format is an important part of how a listener engages with it: a serialised story, non-fiction or otherwise, demands sequential listening of all episodes. A podcast in which every episode is an interview with a different person or is a variation on a theme invites cherry-picking episodes.

A user's cherry-picking from a subscribed show is a valuable tool in mapping a user's interest through episode content because it is reasonable to assume that users saw most of the episodes in the show they subscribed to. Another metric for episode preference would be patterns of episode selection from those offered via the Entale app. However, the Entale dataset does not provide complete episode lists for each show or episodes offered/selected, making it hard to say with certainty what episodes were actively chosen or avoided.

4.1.2 Single Listens

A single episode listen without a subsequent subscription to the show could indicate the user's dislike of the episode. It could also be an episode that met a specific informational need. Multiple episode listens without a subscription point to a lack of intent or value alignment with the episode's content or style, indicating a potential "guilty pleasure" listen. Of course, this is only the case if the user has successfully subscribed to other shows and seems confident with the technology. A series of single episode listens might mean the user is searching for the right podcast in a particular topic or genre. If their search is successful and results in a show subscription, this provides valuable information on their preferences as they test and reject episodes. Date of subscription is not part of the Entale dataset so it's not possible to check the sequence of listen and subscription.

4.1.3 Overlaps

Overlaps in subscriptions and listens show alignment between user's desired behaviour and their reality. In these cases a lot can be learnt from the timing of listens. If listening is a ritual, a user will listen close to the release date of an episode. A quick succession of the same podcast suggests the user is 'bingeing' content, perhaps hooked on a storyline. Again format is an important confounding variable, clarifying the level at which listeners are engaging. Bingeing a non-serialised podcast with long episodes suggests a strong interest in the content itself, whether its the topic, presenter, their guest selection or its sub-culture. Bingeing a serialised murder mystery with short episodes is more likely to suggest an interest in the form, tone or style of the piece.

4.1.4 Discovery Preference / Bravery

Users' preferences for podcast discovery could help create recommendation systems that support their existing habits. For example, a user who listens to a wide range of podcasts occasionally might want more eclectic recommendations than a user who listens religiously to a few podcasts on a particular topic. Willingness to experiment with content and go down a rabbit hole is likely to vary between individuals.

Topic or content preferences may interact with other features such as tone, format and audio quality. Each individual has a complex interplay of interests and preferences. For example a user may prefer content in the category of 'Religion and Belief' to be delivered in an informal, light tone, while expecting their 'Science and Technology' podcasts to be factual and serious. They may be willing to sacrifice audio quality to get access to clear information on their niche science interest, too.

4.2 Derived User Datasets

Based on the above observations, we construct the following derived datasets that are used as inputs to collaborative filtering among other analyses. These derived datasets may be modified as user behaviour is better understood and more accurate, granular assumptions can be made.

4.2.1 User-Show-Listen matrix

In this matrix M, the rows represent users and the columns represent shows. The entries are binary, with $M_{ij} = 1$ if user i has listened to at least one episode of show j, and is 0 otherwise. Only users who have listen to at least 5 different shows are included.

4.2.2 User-Listen-Subscription-Show matrix

Each user's episode listen was tracked back to a show. If a listen was paired with a subscription, it was added to previous listens of that show. If a listen occurred without a subscription only once, it was labelled with a -1 as this solo listen likely signals dislike. However, if there was more than one listen without a subscription listens were counted positively, potentially identifying users listening as a guilty pleasure. Shows that were subscribed to but had no listens in the 12-month span covered in the dataset were labelled with a zero. Any shows that had neither listens nor subscriptions were labelled with a 'NaN', and were not used to fit any subsequent models. Once again, only users with interactions with five or more shows were included.

4.2.3 Listen-Subscription Matrix

Leveraging the logic of the *User-Listen-Subscription-Show matrix*, we construct a user-episode matrix M with three possible values for entries m_{ui}:

- $m_{ui} = 0$ if user u did not listen to episode i
- $m_{ui} = 1$ if user u listened to episode i, but is not subscribed to the show
- $m_{ui} = 2$ if user u listened to episode i and is subscribed to the show

As above, the assumption is that episode listens of subscribed shows indicate a stronger preference than a possibly isolated listen that did not lead to subscription of the show.

5 Models

5.1 Topic Modelling

Topic modelling is an unsupervised machine learning technique which takes in a set of documents as input, detects patterns in combination of words and phrases present in them, and clusters the combination of words or similar expressions that describe or characterise the documents. For this recommendation system task, the documents are in the form of transcripts or descriptions of the episodes. And with the help of topic-models, we can extract patterns from these transcripts and descriptions of the episodes and have a succinct representation of episodes. These representations can further be used for other downstream tasks.

One of the research directions that we explored was the topic modelling of the podcasts to recommend new podcasts based on the similarity to the topics that have been to a user previously. Three different topic models were investigated: *Latent Dirichlet Allocation* (LDA) [2], *Hierarchical Dirichlet Process* (HDP) [3, 4] and a network-based approach, *Hierarchical Stochastic Block models* ([5]). In each of these topic models, it was possible to encode the podcast into a feature space that we could use to make podcast recommendations.

5.1.1 Network-based approach

The challenge owner has done a pilot experiment where name-entity co-occurrence in podcasts is used as a guidance for podcast recommendation. Specifically, for a given user, we can recommend those podcasts that consist of name-entities which have been mentioned in the user's listening history. A natural extension of this strategy is to consider the *community structure* [6] of the entire podcasts-entities network. Inspired by the work in [7], we firstly construct a *bipartite* network of the podcast episodes and the named entities mentioned in each episode. Then, we obtain the community structure in this network by implementing the statistical inference approach based on the family of Stochastic Block Models (SBMs) [8]. Under the assumption that whether a name-entity is mentioned in a podcast is dependant on their semantic

topics, the community structure in the corresponding bipartite network serves as an approximate of the topics assignment of podcasts and name-entities. The implementation of the network-based topic model is done via the `graph-tool` library [9].

5.1.2 Latent Dirichlet Allocation (LDA)

LDA is one of the *admixture* or *mixed-membership* models, where a document does not necessarily belong to a single cluster but is characterized by a group of clusters (topics in this case), and with varying proportions. LDA, proposed by Blei et al. [2], is a probabilistic admixture model that assumes independence between different clusters. We used LDA on transcripts and the descriptions of the episodes with different number of clusters (k) that is provided to the model apriori.

The LDA models were implemented using the `tomotopy` package in `Python`. While there exists many other packages that implement various topic models, we found that `tomotopy` had a wide range of different models implemented and we found it very easy to make slight changes in our code to implement a different model. Due to the time constraints of the challenge, we only implemented LDA and HDP with `tomotopy`. However, the package also includes other major topic models including Supervised LDA [10], Multi-Grain LDA [11], Hierarchical LDA [12], Pachinko Allocation [13], Correlated Topic Models [14] and many more.

5.1.3 Hierarchical Dirichlet Process (HDP)

Hierarchical Dirichlet Process (HDP) is an another admixture topic model [15] that we used for understanding or representing the episodes using the transcripts or descriptions. Unlike LDA, which can be thought of as the finite counterpart of HDP, in HDP there is no need to specify the number of clusters apriori. In HDP, each data point or document (transcript or description in this case), is modelled as a mixture of components and the number of components is inferred automatically by the model. Even if each podcast is described or categorized with a particular (or finite) set of categories given by Entale, we believe that each episode of a podcast can possibly span across *multiple finer topics*, which would increase as

the number of podcasts or episodes increase in thus Entale system. And thus, deciding upon a predefined finite set or number of topics might not be appropriate to describe these episodes from the shows. This is where HDP plays a crucial role and helps in getting a succinct representation of the episodes for further downstream tasks.

We used the implementation of the HDP model from the `tomotopy` package in `Python`. We observed that after running the HDP model on the description data, for almost all the documents, the topic distributions obtained were mostly concentrated only on one or maximum two topics. HDP being a non-parametric model is an important model to consider for topic-modelling of the episodes, and would be one of the near-future aspects to explore and get a better understanding of the topic-distributions obtained using the model. Also, HDP model should also help in understanding the correlation between the categories mentioned by Entale and the categories obtained from the topic-model.

One of the topic models that we think is worth exploring for the podcast system is the Hierarchical LDA [12], which describe the documents with the help of topics and these topics are organized into hierarchy. This would help to describe the documents or episodes using coarser to fine grained set of topics at different level of the hierarchy of topics.

Before going ahead with the next section, here we describe the broader steps that help us use topic models for podcast (episode) recommendation.

1. Obtain embedding of the episodes using the topic models
2. Obtain an embedding of the user (in the same vector space as in the first step) using the user-listen history. Let's call this as *user-embedding*
3. Find episodes for recommendation in the *neighbourhood* of the obtained *user-embedding*.[1]

[1]Note that, the definition of the *neighbourhood* and how we select the episodes for recommendation from the *neighbourhood* matters depending on the way we want the user to explore the episode space.

5.1.4 Data Cleaning

When building our topic models using the transcription named-entities or the description named-entities, we simply took the `span` column in the data frame. However, when building a topic model using the episode transcriptions or descriptions, a considerable amount of data cleaning was necessary. To clean the text data, we did the following:

- Punctuation removal
- Tokenization - the process of splitting strings into a list of words
- Removal of stop-words - words such as “i”, “me”, “ourselves”, “you”, “you’re” will be removed since they are not very meaningful for a topic
- Lemmatization - the process of reducing a word to its root form
- Removal of numbers
- Removal of terms that occur very frequently across all documents - where the rationale being that they are not particularly interesting words if they occur in a lot of documents For instance, words such as “promo”, “podcast”, “episode”, “sponsor” are not particularly interesting since they occur in most episode descriptions and provide little information about the podcast content and topic
- Removal of non-English podcast episodes
- Included bigrams in our bag-of-words - meaning we included pairs of consecutive tokens. This was useful since the podcast transcriptions and descriptions commonly had first and last names. Including bigrams often allowed us to identify first and last names by making sure that they would be paired together

The majority of data cleaning was performed using the following packages in `Python`: `nltk` [16], `spacy` [17], `gensim` [18] and `re` [19].

Figure 12 shows the concentration of words in each topic, where the topics are learned using LDA model with number of topics (k) set to 20 on the episode description data. Also, one of the important parameters that the `tomotopy` implementation considers is `TERMWEIGHT`, which indicates the weight that is given to each word while processing the document. The results shown here are obtained by setting `TERMWEIGHT` to IDF, which is

Inverse Document Frequency, which reflects the importance of a word in the document. One of the important reasons to work with the episode description data is the unavailability of the transcripts for the user listened episodes. However, it would be great to work with other data, specifically transcripts, or combination of other data sources.

5.1.5 Using the topic models to make recommendations

Once we have chosen a topic model and the data that we wish to use to build the topic models, we obtain a distribution of topics for each podcast episode. This enables us to have a representation of each episode into a vector space. In particular, the vector representing each episode in the vector space is the probability distribution over the topics describing the episode obtained using a topic model. Different topic models and using different data (i.e. the transcriptions, descriptions or named-entities) will encode episodes in different vector spaces. We also call this encoding or description of the episode in the vector space as an *embedding*. Once we obtain the embeddings for all the episodes, we can use this as a basis to make episode recommendations.

For a given user, we can firstly look at the episodes that they have listened to. The main step in making an episode recommendation is to use the user's history to represent the user in the same embedding space. There are various ways which we can do this. We focused on three simple methods to combine the episodes to obtain a user embedding:

1. *Concatenate the user's episode data:*
    - For instance, if we were working with episode named-entities, we can combine the named-entities from all the user's episode listen history. And then use the topic model to infer a topic distribution for this combined episode and obtain a vector or embedding that represents the user listening history.
2. *Averaging over the episode embeddings:* Take a simple or weighted mean of the episode embeddings (and normalise)

$$\vec{u} = \frac{1}{N} \sum_{i=1}^{N} w_i \vec{x}_i$$

Figure 12: Word cloud of the topics obtained using LDA with 20 topics on the episode descriptions.

where, w_i denotes the weight given to the i^{th} episode embedding, and $\vec{u}$ is the un-normalised user embedding and $\vec{x}_i$ is the i-th episode that the user listened to. For now, we weight each episode equally. We also normalise the averaged vector so that the elements sum to 1.

3. *Exponential weighted average of the episode embeddings:* To have an "forgetting-history" effect, we take an exponential weighted average of the episode embeddings (and normalise). This weighting scheme associates more weight to the recent episodes and the weights decreases exponentially for the episodes back in time. I.e., older episodes that the user has listened to are given a smaller weight. In particular:

$$\vec{u} = \sum_{i=1}^{N} e^{-i*\varepsilon} \vec{x}_i \tag{1}$$

where, $\vec{u}$ is the un-normalised user embedding, ε is a tunable parameter which controls how much we want to weigh the most recent listened episode (a larger value of ε corresponds to having relatively larger weights to the more recent listened episodes) and $\vec{x}_i$ is the i^{th} episode embedding for the i^{th} most recent episode listen for the user and N is the number of podcast episodes the user has listened to. We normalise so that the vector sums to 1.

Once an embedding is obtained to represent the user within the episode vector space, we can compute the *similarity* between the user embedding and all the other episodes. There are various ways to compare the similarity between vectors. Three similarity metrics we looked at were:

1. Cosine similarity
2. Wasserstein distance
3. Kullback-Leibler (KL) Divergence

The comparison of the different similarity metrics is left for further study. However, we found that the Wasserstein distance and KL divergence to be more appropriate to this project since they are commonly used to compare two distributions.

To make recommendations, we can then compare the similarity metrics between the user and the episodes to recommend episodes that are similar to what the user has listened to previously. In this pipeline, we could simply return the top-n recommendations based on the most similar podcast episodes. To summarise, the recommendation pipeline using a topic model embedding is as follows:

1. Obtain an embedding of the podcast episodes from the topics model
2. Obtain a user embedding by combining their previous episode listens in some way
3. Compute the similarity between the user embedding and the episodes
4. Return the top-n episodes that are the most similar to the user embedding

Notice that this pipeline is flexible and can be adapted in a number of ways. As previously mentioned, one crucial point is that the embedding of the podcast episodes can be obtained in different ways including using different topic models on different parts of the data (named-entities, transcriptions or descriptions), using a network-based approach to topic modelling and by using pre-trained models (e.g. BERT and distilBERT [20, 21]). In addition, there are various ways combine the user's listening history and to choose the similarity metric that we wish to use. In the next two sections, we will further explore using a clustering algorithm and dimension reduction algorithm to extend this pipeline further.

Furthermore, we note that the above described method or framework of obtaining recommendations can be augmented with other techniques to obtain a *Rabbit-Hole recommender* depending on the way we want to *push* the user out of their *podcast space*. The idea is to get the top-k' recommendations instead of top-k, where $k' > k$ (k being the number of final recommendations we wish to obtain). We then sample a set of k distinct episodes from the set k' episodes with probability proportional to the *similarity* score obtained of the k' episodes. Sampling would not restrict the recommendations that are highly similar to that of the user's listen history, but would also have some set of recommendations that are not very similar to the user's listen history. We would accept any recommendation from these less similar recommendations should

gradually *push* the user from their closed podcast space.

5.1.6 Clustering

A potential weakness in the method described in Section 5.1.5 is that it can potentially be computationally expensive if the number of podcasts to compare is large, since the method requires the computation of the similarity between the user embedding and all of the episode podcast embeddings. However, it is likely that a user is only interested in a small subset of the episodes. Therefore, in practice, a clustering of the topic-distribution space of the episodes prior to finding the similar episodes could be useful to reduce the number of podcasts to calculate the similarity to. In particular, we applied the k-Means algorithm [22] to the podcast embedding in order to cluster the documents. Then, given a user, we would look at the podcasts that they previously listened to and use that to obtain an embedding for the user. Rather than compute the similarity between the user and each podcast, we can limit this search down to only the episodes in the cluster that the user belongs to.

The k-Means algorithm was implemented using the `sklearn` [23] package in `Python`. To choose the appropriate number of clusters, we used the *elbow method*. In this method, we would run the k-Means algorithm with different number of clusters k and compute the residual sum of squares of each point and their cluster centroids. The elbow method chooses the number of clusters where the elbow occurs in the plot of the RSS against the number of clusters. The rationale of this method is that this offers a good trade off between the error and the number of clusters. This elbow method can be carried out in `Python` using the `KneeLocator` package [24]. Alternatively, the *silhouette coefficient* is a metric that could be used to measure the cluster cohesion and separation. This quantifies how well a data point fits into its assigned cluster based on:

1. How close the data point is to other points in the cluster
2. How far away the data point is from points in other clusters

Silhouette coefficients vary between -1 and 1 where larger values suggest that data points are closer to their clusters than they are to other clusters. This metric can be computed with the `sklearn` package and the number of clusters can be chosen to be the number of clusters that maximise the

silhouette coefficient. While this was an option we explored, we found that fitting a clustering model using the elbow method described above was more efficient.

While applying a clustering algorithm to our embedding can help with reducing computation, this can limit the chance that the user can explore a wide variety of podcasts. In particular, clustering can be seen as reducing the pool of potential podcasts that can be recommended to the user by looking at the episodes within the same cluster of the user. We can explore the effect of clustering by asking the recommender to make *sequential* recommendations for a each user. That is, we would take a subset of the user's listening history and look at the recommendations that were given by the recommendation method described in Section 5.1.5. From our experiments, we found that when we used an exponential weighting average to combine the episode embeddings, we were able to see that a user could still move around different clusters. In contrast, when we used a simple mean average to combine the user's listening history or when we concatenated the episodes and used the topic model to infer a user embedding, we would often observe that a user would eventually be "trapped" within a particular cluster.

For example, in one of our experiments, we obtained an episode embedding by building a topic model using LDA on the episode descriptions and specified $k = 20$ topics. We applied the K-Means clustering algorithm using the elbow method which gave us $K = 12$ clusters. For a particular user in the data set with with 7 episodes listened, we looked at making recommendations for the user after each episode listened and taking all the episodes up to that point. Figure 13 shows how this particular user was assigned to the different clusters over time using the three different strategies to combine the user's episode embeddings. We can see while the concatenation strategy (where we would combine the user's descriptions and use the topic model to infer an embedding) and the strategy where a mean was taken over the embeddings, we would see the user eventually being "trapped" in the first cluster. In contrast, using the exponential weighting strategy where more recent episodes were given a larger weight, we would see that the user was being assigned to different clusters over time.

In Figure 14, we have a particular user with 17 episodes listened and

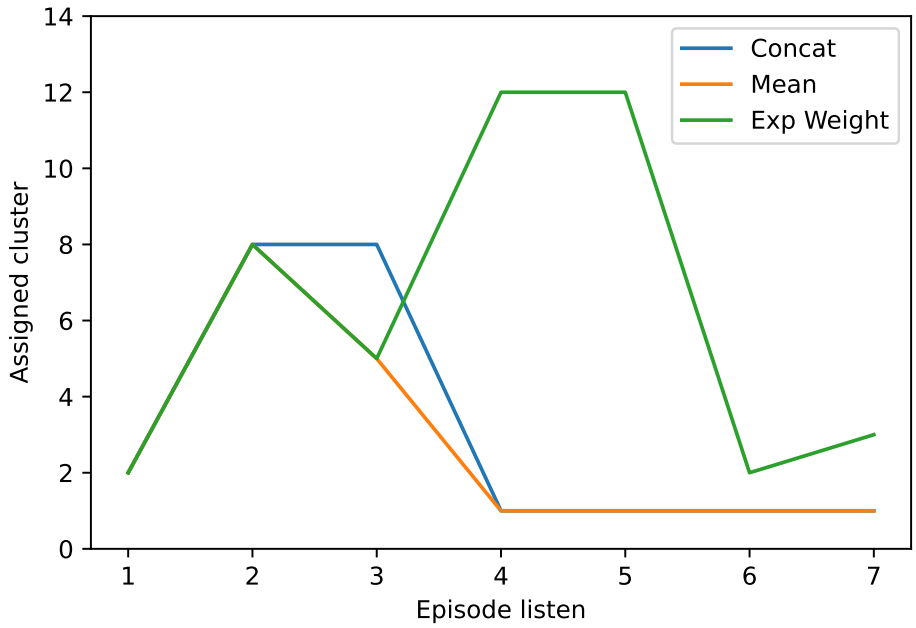


Figure 13: User (with 7 episode listens) assigned cluster over time using different strategies to combine user listening history

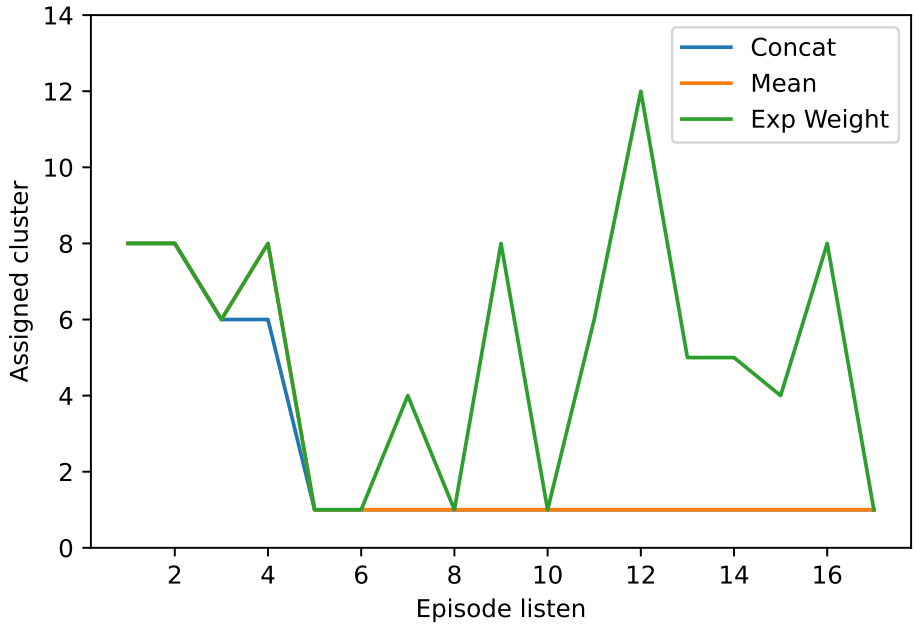


Figure 14: User (with 17 episode listens) assigned cluster over time using different strategies to combine user listening history

again we see the same phenomenon where the exponential weighting strategy allows the user to be assigned to different clusters over time, but the concatenation and simple mean average combination strategies seem to get stuck into a cluster. By having a larger weight towards more recent episodes, there seems to be an element of the recommender "forgetting" the episodes that were listened a longer time ago. How much the recommender "forgets", or places more weight towards more recent episodes, can be controlled by tuning the ε parameter in Equation 1. Since our main objective in this project is to recommend new podcasts that are a comfortable stretch for the listener, rather than to reinforce a user's preferences, it seems from our experiments that the exponential weighting strategy is more suitable since we can still make recommendations for podcasts in different clusters.

5.1.7 Dimension reduction

We also observed that while working with a subset of transcripts from the episodes a non-parametric topic model returned around multiple of 100 topics. And we believe that this could increase as we have more data or more transcripts in the system. Which essentially means that the topic-distribution space would be very high dimensional and would eventually increase the computational complexity for the later tasks like clustering and finding similar or near-similar episodes. And thus, one of the ways to handle this is to reduce the dimension of the topic-distribution space.

5.1.8 Ranking

Once the recommendations are obtained using the topic model and clustering framework, one of the important aspects is the ordering of the recommendations when displayed to the user. One of the easiest ways to rank these recommendations is just on the basis of the similarity measure used (i.e. highest similarity first). However, these recommendations are just based on the topic model and thus capture only the topic related features. One of the ways to add in the diversity in the recommendations is to take into account the user network neighborhood. Basically, if we want to have some network based recommendations for a user u, one of the ways, is to find the set of users v_i who have listened episodes that the user u has listened before and take consider the episodes from users v_i as recommendation that user u has not listened before. Once all these recommendations are obtained, these could be ranked or ordered considering other features in each of the recommendations such as *–sentiment, tone, format of the podcast, style of the speaker, tone, speaker diarization, etc.*

5.2 Network Embedding

Another way of deriving embeddings for podcasts is to use network embedding methods. While there are different networks that we could build from the data, we only investigated bipartite network of episodes and their named entities so that the embedding represents semantic meanings of episodes. We used an unsupervised method named node2vec [25]. Node2vec is a random walk-based representation

learning method that embeds two nodes in the network close if they have high probability of co-occurrence in random walks traversing the network.

The advantage of using this method on the bipartite network of episodes and named entities is that it will embed both of them in the same space. Having so, we could also find related episodes to a named entity and have a relevance metric between episodes and named entities. This property will enable us to recommend episodes based on an specific named entity, and we also use it to find most relevant named entities of a podcast.

To generate the network, we use both description and transcription entities and increase edge weight between the episode and named entity by 1 for each occurance of the named entity in the episode. We could investigate weighting named entities from description and transcriptions differently. Figure 15 shows how the named entities and episodes are placed in the same embedding space.

Figure 15: 2D visualisation (using PCA) of node2vec embeddings for episode named entity network.

Looking at episodes only and their categories, we could see that while the embedding space does not seperate different categories completely in two dimentions, the embeddings have information about the categories (Figure 16 and 17).

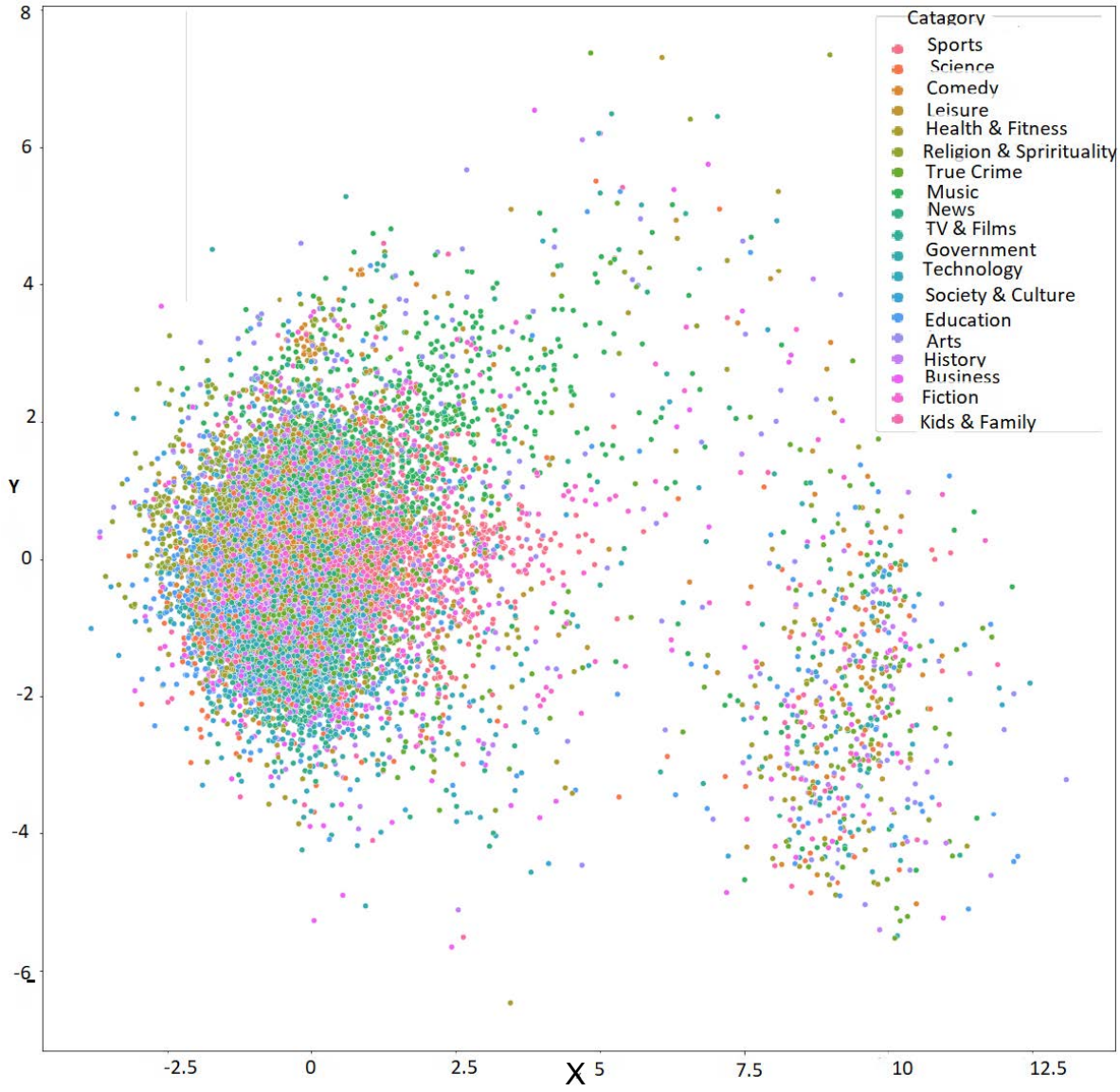


Figure 16: 2D visualisation (using PCA) of node2vec embeddings for episodes coloured with the episodes' categories.

Figure 17: 2D visualisation (using PCA) of node2vec embeddings for episodes in different categories.

While we only investigated the episode named entity network to keep the embedding semantically meaningful, one future direction would be to also consider co-listen network as another network layer that would help the method finding similar episodes based on their user base.

5.3 Rabbit Hole

One of Entale's aims is to generate podcast recommendations that expand on a topic discussed within an episode. These "rabbit hole" recommendations take a user on an unexpected journey through interlinked topics. Rather than relying on a holistic measure of similarity between episodes, provided by modelling the episode's contents overall, a rabbit hole approach focuses on an overlap in content. Specifically, a topic mentioned briefly in the source episode is the main topic of the next. rabbit hole recommendations make it easy for users to be curious and expand their interests, two key goals for Entale.

Recommending episodes based on similarity can lead to circular recommendations: the original episode appears as a top recommendation after only the first or second move from the original recommendation. The rabbit hole method can be combined with similarity-based methods to help push users into new topics or podcast genres. The rabbit hole is not just about one recommendation but a sequence of recommendations. It can be imagined as a wormhole that moves a user into a new part of the episode space.

5.3.1 Transcript-based Rabbit Hole

The rabbit hole pipeline takes an episode as input. In the case we discuss here, this input is the episode a user most recently listen to. The first step is to explore the transcript of the input episode. We do this by obtaining the episode transcription entities; counting how many times these entities appear inside the transcript; and omitting any entities which appear less than twice and that are too common[2]. The reason behind this being; we do not want to accidentally follow a rabbit hole based on an entity which is

[2]This procedure is completely modular and can be updated in the code to include any extra filtering, which will surely be done in further work.

only mentioned once, for example as a humorous comment; and we also want our rabbit hole to be following the main topic of the podcast. In the pipeline, we have now transformed our input episode into a list of potential entities for us to form rabbit holes from.

The following step involves creating links between the list of candidate rabbit hole entities and other episodes in the podcast ecosystem. To do this we directly search the training data (such as the description entities of each podcast), to find podcasts which have mentioned these entities. The result of this is a subset of episodes that have some direct connection to the candidate entity links found from the most recent listen of a user.

At this point, the question remains - how should these candidate episodes be ranked? There are a number of options here and further work will involve investigating and evaluating the success of each one (see Section 6):

- **Similarity to the most recent listen:** could provide a useful ranking which sorts the candidate linked episodes based on how similar they are to the most recent listen. This is essentially as though we are feeding the input episode back into Pipeline B, but with a subset of the original episodes to make recommendations based on.
- **Similarity to *any* past listen:** This is perhaps more in line with the rabbit hole philosophy would be to avoid too much focus on the recent listen, but rather ensure that the recommendation is at least similar to something the user has, in the past, enjoyed.
- **Similarity to the user embedding:** is also a viable option which tends to prefer the episodes which are similar to the users previous listening history while still containing the desired rabbit hole link. This is therefore the ”safest” of the options, however this may not be the desired final goal of the rabbit hole.
- **Strength of a specific direct link:** based on the number of times this directly linked entity occurs in the candidate episodes. However this may be biased towards always recommending episodes with longer descriptions/transcriptions, due to it being more likely that these entities occur multiple times. Potentially this could be done but would have to be normalised by the length of the episode text.

In our implementation, we conducted most of the work using the similarity to the *most recent* listen, since this achieved the rabbit hole effect we were hoping for and didn't leak in any user based preference that would potentially pull the user back to their comfort zone unnecessarily.

As with the other pipelines, all of the similarity-based ranking options above can utilise different embeddings, to approximate different notions of similarity. We believe that one particularly promising embedding type to pursue in this regard are embeddings based on *tone*, style, and 'type' - since this is likely to be in line with the goal of recommending topics which are very different in topic from the user's typical listens, but which they would nonetheless enjoy.

Limitations: The lack of data overlap caused an issue in testing the rabbit hole productively, since we not only require the description entities for an episode, but also the transcription entities and full transcriptions for each episode.

Another potential limitation of the rabbit hole is falling into a spiral or loop. Since the algorithm focuses on topic based on sometimes just a single entity (that has been mentioned a number of times), there is the potential for the recommendations to lead the user into an especially niche area. Niche areas themselves can result in a spiral, if the podcast doesn't include a very wide range of topics (i.e. entities are all belonging to the same topic). One experiment related to this is proposed for future work (see Section 8.3.

5.3.2 Semantic Network Rabbit Hole

The Semantic Network Rabbit Hole is different from the previous pipeline in its embedding and its use of named entities as input, though it operates with similar principles. After selecting some entities from transcriptions, we combine them with named entities from the episode description, as there are usually only a few for each podcast (on average less than 4 named-entities per description).

As discussed in 5.2, a network embeddings method embeds both episodes and entities in the same space. We use network embeddings to select only a few named entities from the transcription entities provided by Entale. Because the number of entities provided per transcription can

be large, the most relevant need to be chosen. Two main approaches for selecting named entities would be:

- Selecting most similar named entities to the last podcast or user's embedding.
- Selecting few named entities that distribute diversely in the embedding space.

We have implemented the first one, and selected two most relevant entities from transcripts by finding the entities most similar to the content of the original podcast. The second approach to selecting transcript entities could be implemented by first clustering the named-entities and selecting one from each cluster, but will be explored in future work. Having these named entities, we could then rank episodes mentioning each named-entity by either method mentioned in 5.3.1. For our current ranking, we use network embeddings and similarity to most recent episode a user has listened.

To wrap up, the general pipeline is as follows. First we select two named entities from episode's transcript that have embeddings close to the episode's embedding. Then we add description entities to obtain a list of entities that we will use for recommendation. Finally, for each of the entities, we select top-k (Currently 4) ranked episodes that mention that entity based on their similarity to the episode.

5.4 Collaborative Filtering

5.4.1 Framing the prediction task

The models described above are content-based approaches that consider primarily the podcast text (including topics, tone, and style). In this section, we describe our attempts at using data on user preferences to make recommendations. Even though we present them as separate sections, it is worth mentioning that there is significant overlap in the information used by the topic modeling and collaborative filtering pipelines.

Making podcast recommendations can be framed as a prediction task. We aim to forecast which recommendations will be seen as enjoyable and

keep a user engaged. Based on our data availability, we identify three avenues for listen prediction. In considering any of these approaches, it will be important to remember an essential feature of the data: that episode listens - the user interactions to which we have access - are *implicitly* indications of preference, rather than explicit expressions of preference, as would be the case if the data include podcast ratings, for example. This changes what we can infer from the data, in particular the 'negative' observations.

- **Time Series Forecasting**. For each user, we have a time series of the episodes they listened to. Reshaping the data into a *(user, time, feature)*-cube, where features are episodes as well as user, episode, or show meta-data, we can apply several time series forecasting techniques, such as recurrent neural networks, to predict the likelihood that a user will listen to certain episodes in the future. Given the short time series and time constraints of the challenge, we leave this approach for future study.
- **User-episode pair classification**. Ignoring the time stamp, we can train a supervised machine learning model to predict the likelihood that a user will listen to a given episode. We classify each user-episode pair according to the preference rank. In this scenario, input features may again be user, episode, or show meta-data, as well as previous listens and engineered features, such as the most prominent podcast category, whether the user is subscribed to the episode's show, the topics covered in the episode, or the user's likelihood to listen to popular episodes. There are a range of appropriate candidate algorithms for this task, including decision tree ensembles, kernel machines, or neural networks. Applying methods to determine feature importance, such as Shapley values [26], may also help improve interpretability of recommendations. The limitations associated with reproducing user behaviour apply to this task. Again, this approach is beyond the scope of this challenge and is reserved for future work.
- **Collaborative Filtering**. A commonly-used approach in recommender systems is collaborative filtering, which considers user-item interactions as a collective and may be considered a generalization of the classification approaches described above

[27]. It uses both the similarity of user behaviour and the similarity of the items that will be recommended (in our case, either shows or episodes). The objective is to predict the unknown entries in the user-item interaction matrix, based on what is known about users and episodes/shows as a whole [27, 28]. This lets us determine whether a user would listen to an episode (show) if recommended. Collaborative Filtering is the primary predictive approach we use to incorporate the user-item matrix. Detailed methods and results are presented below.

5.4.2 Methods

Matrix Factorization using Stochastic Gradient Descent

We factorize the user-item matrix using stochastic gradient descent (SGD). This method belongs to the class of latent factor models and is a commonly-used approach to collaborative filtering. The methodology is well-explained in [27] and [29].

We begin by using as input the binarized show-listen matrix where an entry $m_{ui} = 1$ if a user u has listened to at least one episode from show i. This input was chosen for quick iteration during the challenge, but other inputs, such as the episode-listens matrix may be more appropriate for later implementation, as it allows more granular recommendations. We generate 30-dimensional user and item (show) embeddings by training the SGD algorithm for 3,000 iterations with learning rate $\alpha = 10$, and initial values randomly sampled from a normal distribution with standard deviation $\sigma = 0.5$. We use the mean squared error loss for matrix factorization.

We compute the cosine similarity between users and items in the embedding space. Recommendations are made by choosing the items that are most similar to the user.

Neural Collaborative Filtering

Traditional matrix factorisation approaches to collaborative filtering model user-item interactions as a simple dot-product of their representation. Neural Collaborative Filtering (NCF) was developed to allow for a more complex and non-linear interaction between item and user

representations, and for this function to be learnt simultaneously alongside those representations. This interaction function is (for the most part) modelled by a standard multi-layer perceptron (feedforward, fully-connected neural network), and the model is trained by standard gradient-based methods. See the original paper [30] for full details of the algorithm.

One of the key decisions to make when training this algorithm on implicit preference data is the ratio of negative to positive observations included in the training set. On the one hand, not listening to an episode is more likely to be because a user never saw the episode than that they decided they were not interested in it (given the vast number of podcasts), and so it would be a mistake to try and fit all of these negative observations. On the other hand, fitting only positive listens is trivial - just score/recommend every podcast highly - but this is even more unhelpful as a recommendation system. We strike a balance by subsampling the negative observations on which we train. In particular, we experimented with both a 4:1 and 10:1 negative:positive sample ratio.

Convolutional Matrix Factorisation

One additional method of interest was that of Convolutional Matrix Factorisation (ConvMF) [31]. This method functions by constructing a conventional L_2 regularised loss function, with direct MSE for the target values only incorporated for known values (*i.e.* not weighting null values in the matrix), and item embeddings calculated using a convolutional neural network, that thus allows inclusion of additional item metadata.

In our work, we consider shows to be the items, and the target values are the corresponding user-show position in the user-listen-subscription-show matrix. Item metadata used was as in the original paper, *i.e.* text – the text chosen to represent a show was the concatenated descriptions of all episodes of that show in our dataset. To convert to suitable input for the neural network, an embedding for this text is required – we initialise with pre-trained 200-dimensional Glove embeddings [32], which are then further tuned during inference. Optimisation then proceeds by a combination of backpropagation for neural network weights, and regularised alternating least squares. We updated the author provided Github package here for Python 3 and the current version of `keras`, and

obtained preliminary results, but further experimentation with hyper parameters (*e.g.* embedding dimension for users, items, and text, as well as regularisation parameters) is necessary, alongside possibly the choice of loss function, and the inclusion of additional features.

5.5 Other feature engineering

5.5.1 Sentiment analysis

One avenue of exploration conducted early in the study group was whether sentiment analysis may be a useful additional feature for later recommendation pipelines. We investigated various options, and quickly implemented pre-trained VADER sentiment analysis [33] available through the NLTK python package [16]. However, we immediately came across the issue of the limited number of episode transcriptions available in the dataset – results when applied to the descriptions alone were typically poor, as may be expected as the format is very different to both human speech, and the social media posts that VADER was trained for.

Due to this lack of suitable data, we did not proceed further along this avenue, though we still believe this may likely be an informative way of quantitatively capturing something important about the tone of the podcast that users may be particularly interested in – *e.g.* whether they are positive or negative with respect to a certain issue being discussed. In future, alternative methods that are designed (and ideally pre-trained) on a large corpus of human discussion are likely to be more suitable – we comment on this briefly in Section 8.4.2.

5.5.2 Tracking User Trends

A potential addition to the overall system which could feed into the other pipelines is to track population trends from for example, social media, news articles etc.

A recommendation system that also takes this into account could provide extremely relevant recommendations aimed at the user based on trends in the real world. Some of this may be captured in the collaborative filtering approach. However this will indefinitely incur a time lag between episodes

gaining enough listens for them to be recommended to other users. By having a system that preemptively predicts the behaviour of the overall population inside the podcast ecosystem, this would result in up to date recommendations. For example, if a blockbuster film is gaining a lot of attention in the news and on social media, the recommendation system could give preference to episodes related to the subject.

6 Evaluation

Optimal evaluation would consist of comparing a new recommender system against the current baseline system through a user A/B test. As this is not possible, this section considers three approaches that each provide a different evaluation perspective and complement each other, rather than giving a complete evaluation score on their own. We consider (i) coherence (section 6.1), (ii) reproducing preferences (sections 6.2 & 6.4), and (iii) human inspection (section 6.4). Each of these methods was used to evaluate a portion of the full pipeline. These results may help inform which models are presented to users for A/B testing, but should be interpreted carefully in light of their respective limitations.

6.1 Baseline strategies

Once a recommendation is in place, it is easy to assess its efficacy through the associated downstream task. In this Entale podcast recommendation project, there are three possible baseline strategies we can use for comparison.

- Name-entitycomparison.co-occurrence: Entale has started pilot experiments in which users are suggested to look at podcasts with name entities that appear in their listening history. The underlying assumption is that the co-occurrence of name entities is a good approximate to the similarity between podcasts. Preliminary results show that this strategy tends to work well. However, it is obvious that this approach only considers the direct neighbours of each podcast, ignoring all the other relational information in the name-entities and podcast networks.

- *Entale discovery stream*: One way a user can explore new podcasts in the Entale app is to look at the ranking chart of podcasts by category. Even though it is natural to let users explore themselves, this approach can be improved from the following perspectives. Firstly, users might not always know exactly what they like to listen from the app. Part of the reason for building recommender systems is to help users to make discoveries of potential interest, even those they have not been aware of. This approach might not be as good

as customised methods since for every user it presents the same recommendation result, regardless of user's listening history.

- *User listening history*: Another simple strategy is to recommend podcasts that are similar to users' recent listening history, e.g. recommending episodes of the same show or same categories. Despite its simplicity, this strategy provides customised recommendations for each individual user. Nevertheless, this approach tends to wander around user's explored region and lacks the capability of creating a nice 'rabbit hole' experience.

6.1.1 Characterisation by Recommendation Meta-Features

One strategy is to characterise a new method by qualitatively comparing it against the three baseline strategies above. For example, to compare the network-based approach with the discovery stream strategy, we can look at the category distribution of each topic community, as shown in the Fig. 18. Except for topic 10 which is dominated by Sports category and topic 11 which is mainly about TV&Film, the category distribution of each topic community is quite diverse. In a similar vein, we can study the name-entities co-occurrence between a query podcast and the recommendation generated from this query, as well as the accuracy of matching user's listening history.

The overall idea is to understand the ways in which the recommendations produced by the various algorithms differ from those produced by baseline approaches. Whether those differences are beneficial would be a human decision or decided by testing.

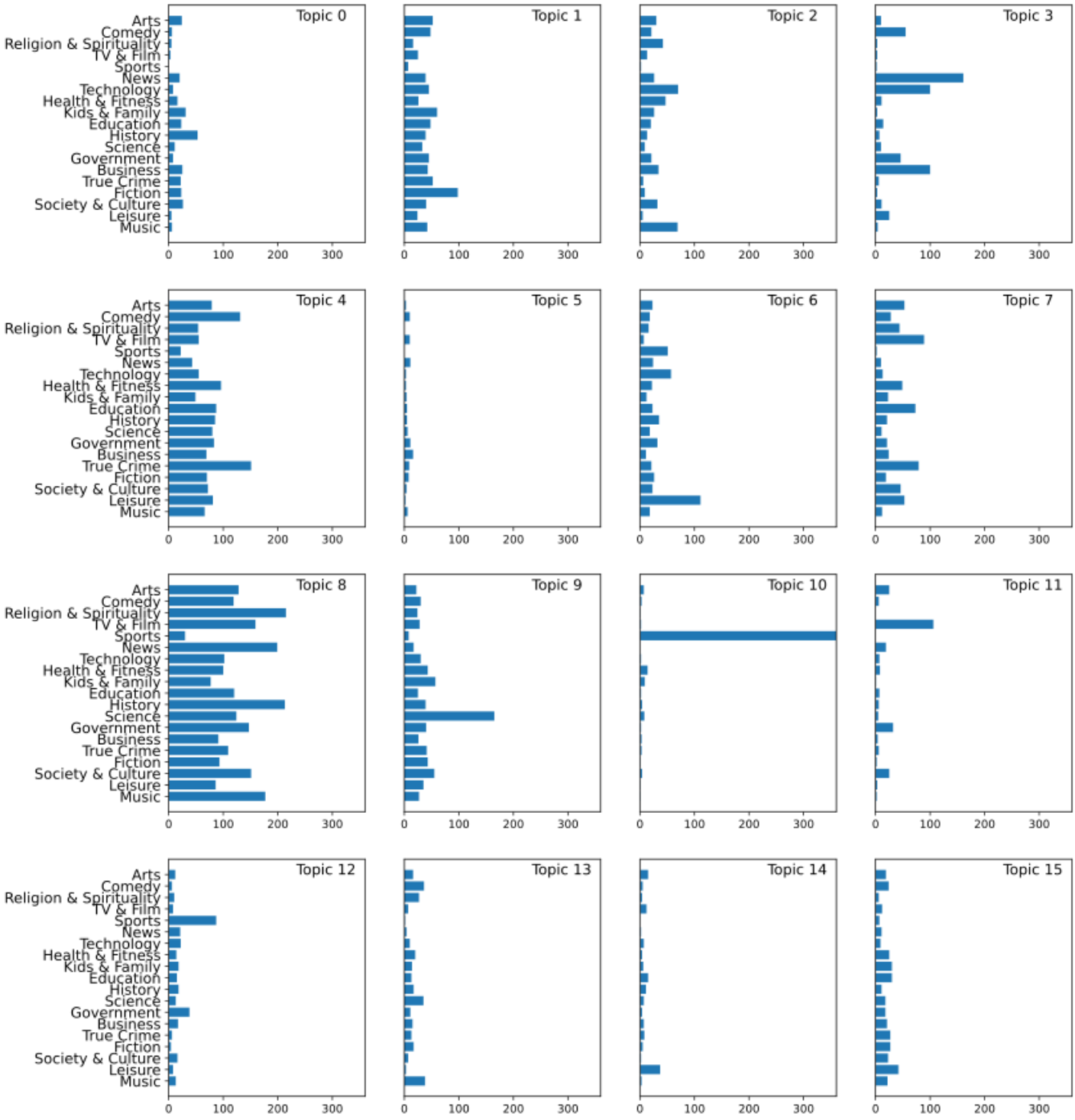


Figure 18: Podcasts are assigned one of 19 primary thematic iTunes categories by the publisher. Following the discovery stream approach, user will select from a pool of episodes of the same category. In comparison, a recommendation based on topic communities will look at candidate podcasts from diverse categories.

6.2 Evaluation Metrics

A successful novel recommendation system should achieve better performance than aforementioned baselines methods. However, providing an estimate of the performance of a recommendation strategy before its deployment is hard. This is for a number of reasons. Firstly, while a prerequisite for a good recommendation system is its ability to capture existing user preferences, by definition its goal is to recommend podcasts which a user *has not yet listened to*, thus inducing new listening patterns. This highlights the dynamic nature of the task of making recommendations - users' behaviour can be changed by the options presented to them - and explains why judging a recommender by its ability to predict existing listens/not-listens alone is mistaken. As a result, it is difficult to conclude which system produces the best recommendation until we can launch some A/B testings. Nonetheless, there are three approaches we propose as preliminary evaluations of the various recommenders described in Section 5.

6.2.1 Scoring vs relative ranking of past listens

One possible approach to evaluating a user-based recommender system is to provide the partial listen-history of a user (e.g. omitting the user's most recent listen), evaluating whether or not the omitted episodes appears in the top-k recommendations produced by the system. The problem with this, however, is that it assumes that the 'ground truth' optimal or near-optimal recommendation for the user would have been that past listen, this is not necessarily the case. Indeed, this data was collected in the absence of a recommendation system, the whole premise of developing such a system is the idea that there exist alternative episodes which the user would enjoy and appreciate, but to which they have never been exposed (the ecosystem is so large they will only explore a tiny fraction themselves). Thus in reality the ground truth top-k recommendation could perfectly plausibly *not* include the user's most recent listen, and optimising for this metric would in fact compromise the goals of the recommender.

There is one general principle of the top-k evaluation approach which is important and should not be disregarded: the fact that we want concrete

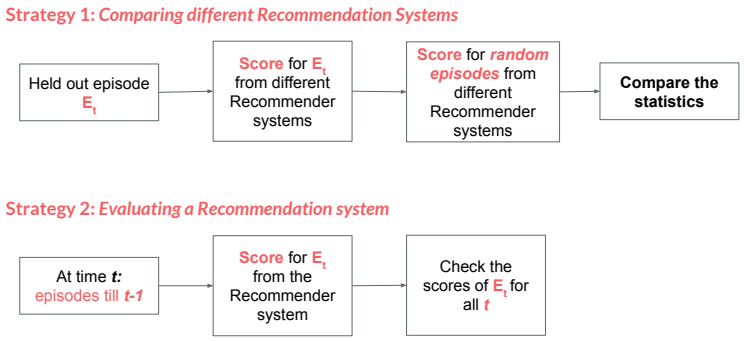


Figure 19: Evaluation Strategies

evidence that our recommender really has learnt the user's preferences.

One approach to providing this evidence is as follows; at an abstract level (Figure 19 Strategy 1), many recommender systems function by *scoring* some set of 'candidate' podcasts. Then ranking them by this score which produces the top-k recommendations. This can be used to evaluate whether the system is able to represent a users preferences. We should expect that the score the system assigns to one of a user's 'past listens' ought to be higher than the score it assigns to a *random* podcast.

This is the underlying principle of the **Hold-One-Out Hit-Ratio** evaluation metric [30], which involves the following procedure. For every listener, their most recent listen is held out as a test set. For embedding based user representations, this means not utilising the embedding of this episode in the user representation, for collaborative filtering approaches, this means not including those observations in the optimised objective function. For each user, we then further randomly sample an extra 99 episodes to which they did not listen to add to their test set. For each user, we then score all 100 of their corresponding test episodes, ranked them by this score, and check whether the episode to which they *did* in fact listen is ranks in the top 10. We call this a hit (or a 'success'), and calculate the success-rate (hit-ratio) across all users. We repeat this process for various methods.

An alternative (and complimentary) approach, which we have not yet explored would be (Figure 19 Strategy 2):

1. For a given user, start by holding out all but 1 of their past listens when constructing the user embedding. Then calculate the scores assigned to their other past listens.
2. Iteratively decrease the number of listen-history episodes 'held out' when constructing the user embedding, and each time again calculate the scores of the remaining held out episode.
3. We would expect that the scores of the held out episodes ought to increase on average as more episodes are used to construct the user embedding.

A couple of important notes on the above.

- The above proposals only evaluate one of the desired characteristics of a recommender system, but an important one.
- These methods are intended for evaluating methods independently, rather than comparing them against one another. The latter is potentially possible but we would need to specify the assumptions built into using them to rank methods.
- We should only be evaluating the system on users with some minimum number of listens (say, 5, for example)
- the above proposals are only for the 'discover screen' set of pipelines.

6.3 Quantitative "evaluation"

We adopt the Hold-One-Out Hit-Ratio approach to evaluation of those recommender pipelines for which it is suitable (in particular, those which have the ability to assign a score to any given user-episode pairing. In addition to considering collaborative filtering approaches, this evaluation approach was built into into the overall API framework. However the results were not ready in time for this report. This will therefore be a simple and immediate extension to the work that Entale can investigate further. This evaluation function is described in detail in section 5.4 and implemented for some collaborative filtering methods.

6.3.1 Collaborative Filtering

For data sparsity reasons, we evaluated our collaborative filtering approaches on the user-*'show-listen'* table, rather than focusing on episode listens. However, the code can easily be applied to the episode-data, and the evaluations repeated.

Matrix factorisation using SGD: The untuned baseline approach has a low success rate of only 11.5% which corresponds to 197 of 1714 users, performance which is close to what we would expect from randomly generated recommendations. However, we emphasise that we expect performance to increase significantly with hyper-parameter tuning, including experimenting with different similarity metrics, optimisation approaches, and initialisations.

Neural Collaborative Filtering: The observed hit-ratio for collaborative filtering ranged between 52% and 54%. We consider these results positive, given the low-threshold for minimum listens per user (which we set at 5), and the sparsity of the data generally. At the very least, this serves as a good basis for comparison for other methods, when attempting to measure the extent to which they have capture user preferences.

There are numerous other hyperparameters to tune for the NCF model. We evaluated NCF using the HR10 hold-one-out methods described in Section 6.2.1, for a variety of different hyperparameter settings. We check all combinations of the following parameters: batch size = [128, 256], embedding dim = [10, 30, 50], mlp layer dims = [(64, 32, 16, 8), (64, 32, 16), (32,16, 8), (16, 16, 16), (16,8)], dropout = [0,0.2, 0.5], with learning rate=0.001, with negative sampling ratio in the training dataset of 10:1. As noted in original NCF paper [30], the model quickly starts to overfit, so we implemented early stopping after 4 epochs of non-increasing hit-ratio, and report the top achieved hit-ratio. We found relatively consistent performance across all hyperparameter combinations, as summarised in Table 1.

Mean	Std	Min	Max
53.1%	0.4%	52.3%	54.1%

Table 1: Hit ratio statistic of NCF models across all hyperparameter combinations.

6.4 Qualitative Assessment of Rabbit Hole Recommendations

In the absence of user testing to compare the recommendation systems, the team conducted an informal qualitative assessment of each system's output.

First, a set of users with 10 listens or more available in transcript form were chosen. Users with several transcribed episodes in their history help provide the overlap required to evaluate different approaches to rabbit hole recommendation systems. This small subset of 84 users had many more listens than average to meet this overlap requirement. A user with a listening history of 512 episodes was selected at random from this subset who we'll call 'User Y' (*user_id*:Y9f12DmCDVcYqBjHLoULLQ). A long user history has the advantage of giving listener-based recommendations a large dataset to work with.

Episode 203 in their listens had a transcription and was chosen as the slice necessary for episode-based approaches. This episode is from a BBC flagship show called *Desert Island Discs*. *Desert Island Discs* is an interview-based podcast that invites a celebrity to discuss their life, career and the items they would take to a desert island in every episode. Each episode description features includes information on songs selected by the celebrity as well as a few details of their biography. The episode listened to by User Y features the explorer Steve Backshall, Episode title: Steve Backshall. The description reads:

> Steve Backshall is an explorer, naturalist and broadcaster. His BAFTA-winning programmes bring viewers of every generation closer to nature – from the children's series Deadly 60, featuring close encounters with the most dangerous and venomous creatures on earth, to Blue Planet Live and Springwatch. His interest in the natural world began at a

> young age, after his parents decided to swap their terraced house for a smallholding with goats, ducks and geese. His big break as a broadcaster arrived when National Geographic offered him the post of Adventurer in Residence and he's been taking on the most arduous challenges and toughest environments on earth ever since. He ran a marathon in the Sahara and has swum cage-free with great white sharks. His adventures have also brought him many near-death moments. He broke his back while rock climbing and recently almost drowned while kayaking in Bhutan. Steve is married to the Olympic champion rower Helen Glover, and they have a two year old son and twins born earlier this year.DISC ONE: Beautiful War by Kings of LeonDISC [sic] TWO: The Wind by Cat StevensDISC [sic] THREE: Fake Plastic Trees by RadioheadDISC [sic] FOUR: Even After All by Finley QuayeD ISC [sic] FIVE: I'm Gonna Be (500 Miles) by Ash Cutler and Rachael HawntDISC [sic] SIX: Last Goodbye by Jeff BuckleyDISC [sic] SEVEN: 6 Words by Wretch 32DISC [sic] EIGHT: This Life by Vampire Weekend BOOK CHOICE: One Hundred Years of Solitude by Gabriel García Márquez LUXURY ITEM: A guitar CASTAWAY'S FAVOURITE: I'm Gonna Be (500 Miles) by Ash Cutler and Rachael Hawnt Presenter: Lauren Laverne Producer: Sarah Taylor

The top recommendations produced by the two rabbit hole systems are summarised in Table 2.

Both approaches to the rabbit hole are discussed in detail in Section 5.3. To evaluate the systems, the LDA rabbit hole was run with two different entity prioritisation settings. Setting (tw0) prioritises named entities that are frequent across the corpus, while the other (tw1) prioritises words that are infrequent. In both cases named entities are filtered s o t hat only those that appear twice are considered candidates, and prioritised if they appear in close together in the transcript. The bipartite network rabbit hole is created from the list of transcript and description entities provided by Entale. These named entities have undergone some disambiguation, but there are still instances of the same named entity appearing in different formats, e.g. 'Code' and 'code'. The bipartite network approach is able to make the connections it used to make recommendations

Table 2: Comparison of recommendations generated by different pipelines and models

Pipeline	Topic Model	User Model	Other	Connection	Recommendations
Rabbit Hole	LDA	Exponential Weighting	Frequent words prioritised (tw0)	War, Horror (implicit)	Overall rankings: 1. Meuse Argonne - The Most Serious Business 2. Season 4 - Episode 6 3. Moscow's Communist Dorm 4. MAG 147 - Weaver
Rabbit Hole	LDA	Exponential Weighting	Rare words prioritised (tw1)	Sports, Olympics, Gold (implicit)	Overall rankings: 1. LeBron & AD Overpower Heat In Game 4 2. International Players Anthem 3. Philip Rivers Retires, Brady Beat Brees, and Mahomes Injured
Rabbit Hole	Bipartite Network + node2vec	N/A		Adventurer, Bhutan, Cat StevensDISC [sic]	Top per named entity: 1. AEE (All Ears English) 603: The 6 Signs that Someone Is Flirting with You in the United States 2. 211: No Such Thing As A Photograph Of A River 3. Yusuf Cat Stevens, musician (Desert Island Discs)
Network-based Model	Bipartite Network	N/A	Only episode-based	Unclear	Overall rankings: PqNsFSfFGE1uUYjU4JqZrs AD4NYeekc6bGijhULLUJMi 73BmAPYscgYiPmGh39xekK.

explicit, with the LDA approach does not.

The three approaches produce wildly different recommendations, highlighting the importance of interpretability and user testing to help select successful recommendation systems. When prioritising frequent words, the LDA model seems to have picked up on words related to Steve Backshall's brushes with death as the three episodes recommended have dark themes of war, battle, horror and terror in their episode descriptions. Two of the episodes are history podcasts and one is fiction. When infrequent words are prioritised, however the system recommends episodes that are similar in genre, all discussing sports and the NBA specifically. Because infrequent words are used to find relevant episodes, the recommendations naturally cluster together in a niche field.

The bipartite network approach looks primarily at the named entities in an episode description but also includes two from the transcription that are ranked as the most relevant. It is heavily reliant on the quality of named entity extraction. For example, 'Cat StevensDISC' was extracted as a named entity rather than the artist 'Cat Stevens.' The only episode suggested for this named entity is *Desert Island Discs*'s interview with Cat Stevens, in large part because it's description formatting also produced the error of 'Stevens' and 'DISC' being combined. The wider named entity of 'adventurer' leads to a series of English-language learning podcast episodes, but the entity 'Bhutan' suggests an episode in another genre (comedy / science / general knowledge) that also mentions Bhutan.

7 Limitations

Reproducing listens. Until a recommendation system has been implemented, we rely on listen histories to evaluate the recommendations generated by different models. Reproducing listens may be risky. Any human biases that informed podcast selection may be reinforced through recommendations. The historic perspective may also result in a narrow understanding of the success of different models, and consequently, any evaluation metrics must be interpreted carefully. The rabbit hole approach and other topic- or entity-based recommendations described in our report will help expose users to a broader set of podcasts, provide more complete data, and will eventually reduce the risks associated with reproducing listens. In the meantime, possible technical and human biases should be monitored closely.

Implicit User Preferences. Reproducing listens is especially relevant in the context of the collaborative filtering approaches described in this report. These methods leverage user preferences, which are only implicitly given in our data. This is a major limitation of this part of the recommender pipeline, and should be carefully considered when integrating models into the Entale application. More explicit user preferences may be derived once a baseline recommender is in place.

User testing. The ideal evaluation of the models developed in this challenge would be in an A/B test with Entale users. The modular pipeline allows for easy integration and removal of components, such as collaborative filtering, topic models based on different input text, or the discovery/ rabbit hole approach. Several candidate models may exist for each component and user testing can help identify the best model for each task, possibly considering different user objectives.

Changing Entale user. The user embeddings are based on the current Entale user base. As the user base grows, the embeddings may evolve over time. Some modules in the pipeline can be switched on or off to adjust for new behaviour categories. For example, users may come from different demographic groups, with new niche interests, or with seasonal recommendation preferences.

Monthly listens. When examining the last episode a user listened to, we must sometimes randomly choose between multiple options due to the monthly data aggregation. This can easily be adjusted when moved to production, but should be considered in interpreting the results presented in this report.

Assumptions about user behaviour The meaning of user behaviours, such as subscribing but not listening or listening to an episode multiple times, has not been tested. A stronger understanding of how user intent is reflected in their behaviour would improve the use of listener data. For example, multiple listens of an episode could reflect a strong affinity with a podcast, or it could indicate a user needing to constantly rewind because they got distracted. Interpreting listener behaviour correctly has obvious value for recommending podcasts a listener will enjoy.

Hyper-parameter tuning In general, almost all methods applied include *hyperparameters*, *i.e.* parameters chosen prior to application that affect the output of the method. In many cases, these can have a substantial impact on the quality of the output, and so careful tuning is vitally important. Unfortunately, given the time constraints of the project there was little opportunity to conduct such tests, and so there are some issues in directly comparing the methods applied to one another to choose the 'best' – we leave this to future work.

8 Future work and research avenues

8.1 User Testing

- Different pipelines
- Different weightings/features of user/content
- Different weighting between discovery vs. rabbit hole

Very simple to implement mixture of model recommendations (assuming different models may better recommend *e.g.* 'rabbit-hole' podcasts vs podcasts of general interest to the user), then possibly use subsequent user listens to preferentially choose a model over time, or separate recommendations into different aspects of the product.

Users can also quickly provide low-cost positive and negative signals should certain features be incorporated into the app. For instance, allowing users to choose not to be presented items from a certain show on their homescreen, data from saving/liking a podcast, choosing particular categories and/or topics that they're interested in during an onboarding process *etc.* The onboarding process would also allow provision of higher-quality recommendations more rapidly, as it provides some baseline data for a user without having to wait for them to meet some pre-requisite number of interactions with the platform.

8.2 Gathering More Complete Data

Assuming that user ratings are not available, the most useful data to inform a recommendation system would be:

- Duration of listen
- Time and date of listen
- Transcripts for each episode
- Named entities extracted with information on frequency (or some other metric of relevance) to podcast content at show/episode level
- Other categories a show/episode belong to

- Views/ recommendations that did not lead to listens/ratings

Other possibly useful datapoints could be the platform a user listened on (assuming it will be available on computers at some point), and (again if implemented) whether certain podcasts are placed in a queue together by users. If data collection is costly, in large-scale production implementations it may be more affordable to appeal to data augmentation and bootstrapping approaches to generate synthetic data for model implementation. This could also be tied into differential privacy approaches to ensure user security.

8.3 Markov's Rabbit Experiment

One proposal for a future evaluation of the Rabbit Hole would be to conduct a *Markov's Rabbit* experiment, whereby recommendations are fed back into the system for a number of iterations. By tracking the topic distributions these recommendations belong to, a quantitative measure of how far the Rabbit Hole is traversing the podcast ecosystem could be built up in the embedding space (or by a similar scoring mechanism). Depending on the business goals of this recommendation system, this may not be a very noticeable effect, since in practice, users may tend to only traverse the Rabbit Hole for one or two steps. However for a robust algorithm this may be something to consider, or to turn on if the algorithm detects the user has fallen into a Rabbit Hole spiral.

A potential solution for this issue would then be to find some balance in the ranking of recommendations, either by ignoring direct links that are too closely related to the original episode topic (e.g. comparing to the origin episode description) or by weighting the ranking method chosen to deliver the recommendations to favour more distinct episodes.

8.4 Incorporating other features of tone, style or format

The writing style of an episode description is designed to reflect tone and style as well as content. Features that could help cluster episodes according to tone or style by contents of their episode description were considered, but not implemented.

Tone:

- Punctuation: !, !!, ?!, ?, ??, ...
- Adjectives (amazing, light-hearted, incredible etc etc)
- Verbs: join, learn, explore, share, teach, discuss, debate, introduce, dive
- 'Paralinguistic' cues: Oh, Hey, (words that are in all caps for emphasis)
- Sentences starting with ". But" (polemic/political)
- Sentences starting with "With..." (providing context, signals more complex content)
- Sentences that begin with "Well," (conversational)
- Texting acronyms (WTF, AF, fk etc) - is there a databse of these somewhere?
- Swear words
- "Trigger warning"

Format/Style:

- Interview
- Conversation
- Q&A
- Lecture
- Discussion
- Fiction, Adventure, Imagined
- REVELATORY: Scoop, inside story
- DETECTIVE: Mystery, Victim, Killer, Crime
- EXPERTISE: Professor / Chaplain / Dr / "head of" / "director of"

These features of tone and style, which naturally overlap, could be used to create podcast similarity models equivalent to the topic models

described in this report. They may be particularly useful for filtering recommendations generated by a rabbit hole style recommendation. When a user is being pushed into a new area of knowledge, it may be useful to be able to provide an episode that matches the user's preferred tone (light-hearted, expert-driven, fictional etc.).

8.4.1 Speaker diarisation

Speaker diarisation corresponds to partitioning an audio stream into sections based on the identity of the speaker at that time. While this was not possible during the DSG due to lack of audio data, Entale could apply available methods with lower computational cost to get some idea of (a) the number of speakers active during a podcast, and (b) the proportion of the podcast each speaker is active, as well as possibly (c) the variance of the length of their contributions. We believe that these three features could be highly informative for determining the format of the show, *e.g.* single speaker talking about a topic, interviews, low-key group discussion *etc.*, and that users may display some preference towards such formats beyond just the topic. A long list of recent developments in speaker diarisation is available here – for assessment of their utility in recommendation, the resulting features could either be used directly, or if suitable labelling is first performed, an intermediary format classifier could be constructed.

There has also been recent work using adversarial neural networks (see *e.g.* [34]) to identify speakers across a full audio dataset without pre-labelling, which could be even more useful in terms of better understanding whether users are choosing podcasts with particular people, even if they themselves are not the focus of the discussion.

8.4.2 Audio sentiment analysis

After segmentation into individual speakers, further useful information about the tone of the podcast could be more readily captured. As well as simple features such as pitch *etc.*, there are a variety of pre-trained audio sentiment analysis packages, for instance here. These methods can incorporate both direct audio features such as the pitch and volume as well as those obtained from NLP after transcription. Combining the

resulting estimates of tone and/or sentiment over the full course of the podcast may provide useful features related to the tone for application in other methods. As with format, the justification would be that users often enjoy podcasts based on whether they are *e.g.* positive (say a hopeful podcast about future technology) or negative (like polarising political podcasts).

8.4.3 Deeper semantic relations between podcasts

While we experimented with several different similarity measures, as discussed in Section 5.1.5, there are alternative methods of computing relationships between items, particularly semantic relationships, that might be more suitable. In particular, DistMult [35] has been recently used fruitfully as one feature for a podcast recommendation system [36] – they also implement RNNs as one component, which we touch upon further below, but were not practical thus far given the typically short length of available user-listen sequences in the provided data.

8.5 Alternative methods

8.5.1 Other content-based collaborative filtering methods

Other than ConvMF as discussed in Section 5.4, we did not implement any hybrid approaches for matrix factorisation that included other metadata. As ConvMF may be computationally expensive to scale for the full dataset, other alternatives may be more desirable, for instance [37, 38], though relatively speaking the process should be affordable – on the provided data, the model trained within several minutes on a single NVIDIA Tesla M60 GPU through Microsoft Azure. There may also be benefit to implementing session-based recommendations rather than user-based (see *e.g.* [39]), as particularly for the format of podcasts it is likely that users preferences/interests may vary significantly between periods of listening.

8.5.2 Approximate Nearest Neighbors

Once the embedding for the episodes and/or the users are obtained, one of the important tasks is to understand similar episodes for a given

episode or similar users of a given user. This could be very useful for the recommendation task or at least for getting the candidate recommendations. And given that the data volume might increase in future, doing this similarity based task would be very time consuming and thus having a system that is computationally efficient will be important. Locality Sensitive Hashing (LSH) (`https://www.mit.edu/~andoni/LSH/`) is one of popular techniques to get *approximate nearest neighbors (ANN)* which is used widely from computational perspective. The notion of approximation here is with respect to the distance metric used. And currently LSH is theoretically and practically has been proved to be applicable for a number of common distance metrics such as Euclidean, Cosine similarity, Angular, Hamming, Earth Movers Distance, etc. One of the another advantage of ANN is from the perspective of the "Rabbit Hole" approach. Being approximate might make the user *push* out of its podcast space gradually towards the "Rabbit Hole".

8.5.3 Alternative network formulations

While the only network approach considered herein was to fit an SBM to the bipartite entity-episode co-occurrency network, a wide variety of alternative options are available, and likely desirable. For example,

- Forming a network of episodes, with weighted connections between them if users have listened to those two shows (or more if formulated as a hypergraph) – detected blocks in this case would reveal typical user communities, that may then be useful for subsequent recommendation. Furthermore, the number of steps in the network between two episodes (possibly after thresholding edges) provides some information as to their similarity at least in terms of appeal, without requiring complex further processing.
- A bipartite user-item network (*i.e.* the full user dataset rather than the projection above), where edges might incorporate information about both whether the user has listened to an episode (or show), and whether they have subscribed to a show. Analogous networks are frequently used to provide a baseline means of recommendation [40].
- Dense similarity networks, where edges are defined by some

weighting function based on metadata on the nodes, such as topic distributions inferred for episodes.

- k-NN networks formed from any of the embeddings, *i.e.* where edges between items i and j if item j is within the top k most similar items for i under some scoring function.
- Multilayer network formulations, where all (or some subset) of the provided information is incorporated in different ways in different layers. A variety of approaches exist to interpret the results, from centrality measures to community detection, and many others in between.
- Dynamic network formulations, where the network evolves over time rather than remains static – typically these are constrained to either timeslices (corresponding to a discrete binning/windowing of time) or edge-streaming models, but more data would likely be required for reasonable results. Once again a variety of methods exist, from simple temporal centrality extensions to more advanced statistical models for link prediction.
- Topological Data Analysis (TDA) methods on the resulting networks are also increasingly popular, as they provide useful summary features that can be used as input features for other methods.

We emphasise that major reasons for the rising popularity of network methods are that they are (i) frequently comparably as successful as much more complex approaches, (ii) interpretable, and (iii) often (at least for simpler centrality measures and similar) much less computationally demanding than alternatives. As such, they are certainly worthy of greater investigation than performed within this work.

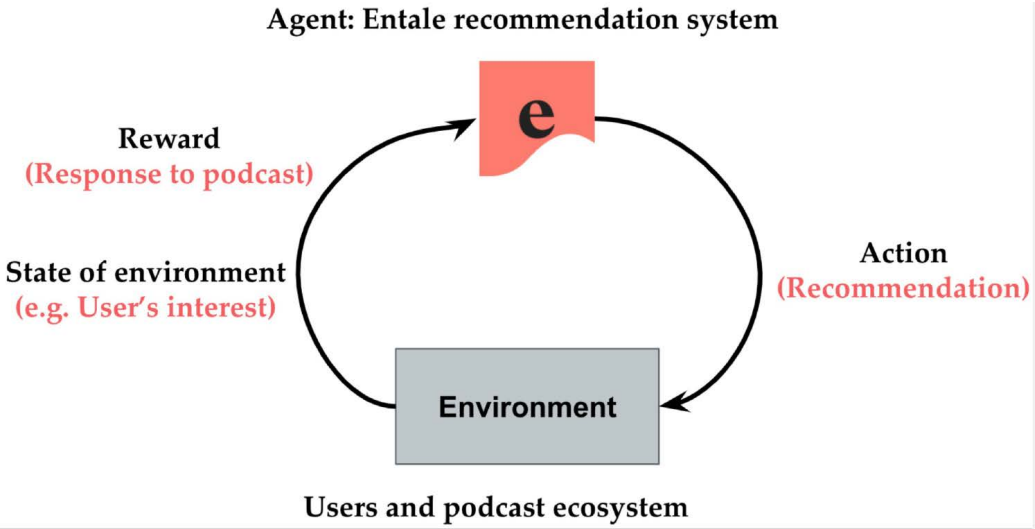


Figure 20: A diagram of RL system for podcast recommendation. Following the classic paradigm of RL, at each time step, the agent (podcast recommendation system) makes an action based on the current reward and state of the environment (users and podcast ecosystem). During the model traning phase, the agent adjusts its decision making strategy such that the cumulative reward (e.g. users' satisfaction) is maximised.

8.5.4 Reinforcement Learning

Most of the methods that have been considered so far assume the recommendation task as a static problem. Nevertheless, it is more appropriate to think of making recommendation as a sequential decision process [41, 42]. Moreover, with the note of the Rabbit Hole spiral in 8.3, it is necessary to take long-term recommendation quality into consideration in the next step. One potential direction to pursue is to formulate the podcast recommendation problem as a *Markov decision process* problem and solve it by *Reinforcement Learning* (RL) methods. Rather than taking the recommendation as a one step action, we like to make a series of recommendations and consider cumulative reward. In the ideal scenario, if a user is close to be falling into a rabbit hole spiral, the model can foresee the decrease in cumulative reward and adjust its recommendation strategy correspondingly.

There have been quite a few successful applications of RL in the field of recommender system [43, 44, 45]. In the typical setting of a RL system, an *agent* learns how to optimise a numerical objective by interacting with the environment it lives in - agent can take actions and the environment will return corresponding rewards. In the case of podcast recommendation, a recommendation system is the agent to be trained and the environment includes users and the podcast ecosystem. Each time our recommendation agent takes an action, i.e. presents a recommendation, a *reward* can be collected from the environment, e.g. users' decision in accepting the recommended podcast (see Fig. 20). The hope is that we can obtain an intelligent recommendation agent after training which can take actions according to different states of the environment such that the long-term reward is maximised.

9 Team members

Jevgenij Gamper is the principal investigator for this project is a final year PhD student at Warwick University working on machine learning algorithms for medical imaging, his supervisor is Prof. Nasir Rajpoot. He has a diverse range of experience in applying supervised and unsupervised machine learning algorithms to a variety of domains: from probabilistic machine learning models for Exo-planet validation; to Bayesian modeling for climate models output down-scaling in industry at Cervest Ltd; to training and deploying deep learning based computer vision algorithms for remote sensing and medical imaging.

The following members contributed equally to the project:

Ryan Chan is a third year Statistics PhD student at The Alan Turing Institute and The University of Warwick, working on developing scalable Monte Carlo methodology for statistical inference with big data. Ryan contributed to the project by working on the data cleaning, implementation of the topic models and taking those models to make recommendations. In addition, he investigated the use of collaborative filtering methods to make user-based recommendations.

Jayesh Choudhari is a Research Fellow at the University of Warwick working on developing algorithms for dynamic data-streams. During his Ph.D. at Indian Institute of Technology Gandhinagar, India, he worked on Probabilistic models and algorithms for information diffusion in social networks. Jayesh contributed to the project by working on the topic modelling and clustering based framework for podcast recommendation.

John Fitzgerald is a third year PhD student in mathematics at the University of Oxford, working on developing statistical models for global networks formed from publication data. He contributed to the project as co-facilitator, helping the team work efficiently together alongside some work on collaborative filtering, in particular the ConvMF implementation and associated data manipulation.

Erfan Loghmani received his M.Sc degree in Artificial Intelligence from Sharif University of Technology. During his M.Sc. he has worked on graph representation learning methods and their applications in recommender systems. Erfan contributed to the project by analyzing user behavior data and network-based recommendation models.

Jamie McGowan is a PhD student at University College London, working on developing Parton Distribution Functions for the proton in the field of Theoretical Particle Physics & Phenomenology. Jamie contributed by developing the user-based, episode-based and rabbit-hole recommendation systems; implementing the various pipelines

Vanessa Pope holds a PhD in Media & Arts Technology for developing computational methods that identify show structure and evolution in live stand-up comedy. Vanessa contributed to user data analysis, the rabbit hole recommendation system and the conceptual integration of separate models.

Ilan Price is a PhD student in Mathematics at the University of Oxford, researching the mathematics of deep learning. Ilan's contributions included data cleaning; developing the LDA topic model, the rabbit-hole pipeline, and the Neural Collaborative Filtering implementation; obtaining the BERT episode description embeddings; and developing the evaluation pipeline for the various user-based recommender systems.

Kirstin Roster is a PhD candidate in computational mathematics with a focus on time series forecasting and causal inference. She was co-facilitator of the project and contributed to the user analysis and collaborative filtering.

Lizhi Zhang is a final year PhD student at the Centre for Doctoral Training in Statistical Applied Mathematics (SAMBa) at Bath. He contributed to the construction of the name-entity and users networks, design and implementation of the network-based recommendation algorithm.

A Show Categories

Every show (and, by inheritance, episode) in the dataset is assigned one of 19 primary thematic iTunes categories by the publisher:

- Arts
- Business
- Comedy
- Education
- Fiction
- Government
- Health & Fitness
- History
- Kids & Family
- Leisure
- Music
- News
- Religion & Spirituality
- Science
- Society & Culture
- Sports
- Technology
- True Crime
- TV & Film

B Named Entity Labels

B.1 Transcription Entity Labels

Label (short)	Label (full)	Examples
PER	Person	Dorothy Day, Paulie Malinaggi,
ORG	Organization	Department of Energy, University of Notre Dame
LOC	Location	London, China, Earth
EVT	Event	Brexit, Olympic Games, Christiams Eve
TLE	Title	Blood on the Dance Floor, Alice in Wonderland, Washington Post
BRD	Brand	iPhone, Land Rover, Coke

B.2 Description Entity Labels

Label (short)	Label (full)	Examples
PER	Person	Dorothy Day, Paulie Malinaggi,
ORG	Organization	Department of Energy, University of Notre Dame
LOC	Location	London, China, Earth
EVT	Event	Brexit, Olympic Games, Christiams Eve
FILM	–	Star Wars, Thelma & Louise, Avengers: Endgame
TV_SERIES	–	Buffy the Vampire Slayer, Game of Thrones
WOA	Work of Art	The Joe Rogan Experience, Django, Google Play

References

[1] Thomas MJ Fruchterman and Edward M Reingold. “Graph drawing by force-directed placement”. In: *Software: Practice and experience* 21.11 (1991), pp. 1129–1164.

[2] David M Blei, Andrew Y Ng, and Michael I Jordan. “Latent dirichlet allocation”. In: *the Journal of machine Learning research* 3 (2003), pp. 993–1022.

[3] Yee Teh et al. “Sharing clusters among related groups: Hierarchical Dirichlet processes”. In: *Advances in neural information processing systems* 17 (2004), pp. 1385–1392.

[4] David Newman et al. “Distributed algorithms for topic models.” In: *Journal of Machine Learning Research* 10.8 (2009).

[5] Tiago P. Peixoto. “Hierarchical Block Structures and High-Resolution Model Selection in Large Networks”. In: *Phys. Rev. X* 4 (1 Mar. 2014), p. 011047. DOI: 10.1103/PhysRevX.4.011047. URL: https://link.aps.org/doi/10.1103/PhysRevX.4.011047.

[6] Santo Fortunato. “Community detection in graphs”. In: *Physics Reports* 486.3 (2010), pp. 75–174. ISSN: 0370-1573. DOI: https://doi.org/10.1016/j.physrep.2009.11.002. URL: https://www.sciencedirect.com/science/article/pii/S0370157309002841.

[7] Martin Gerlach, Tiago P Peixoto, and Eduardo G Altmann. “A network approach to topic models”. In: *Science advances* 4.7 (2018), eaaq1360.

[8] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. “Stochastic blockmodels: First steps”. In: *Social Networks* 5.2 (1983), pp. 109–137. ISSN: 0378-8733. DOI: https://doi.org/10.1016/0378-8733(83)90021-7. URL: https://www.sciencedirect.com/science/article/pii/0378873383900217.

[9] Tiago P. Peixoto. “The graph-tool python library”. In: *figshare* (2014). DOI: 10.6084/m9.figshare.1164194. URL: http://figshare.com/articles/graph_tool/1164194 (visited on 09/10/2014).

[10] David M Blei and Jon D McAuliffe. “Supervised topic models”. In: *arXiv preprint arXiv:1003.0783* (2010).

[11] Ivan Titov and Ryan McDonald. “Modeling online reviews with multi-grain topic models”. In: *Proceedings of the 17th international conference on World Wide Web*. 2008, pp. 111–120.

[12] David M Blei et al. “Hierarchical topic models and the nested Chinese restaurant process.” In: *NIPS*. Vol. 16. 2003.

[13] Wei Li and Andrew McCallum. “Pachinko allocation: DAG-structured mixture models of topic correlations”. In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 577–584.

[14] John D Lafferty and David M Blei. “Correlated topic models”. In: *Advances in neural information processing systems* 18 (2006), pp. 147–154.

[15] Yee Whye Teh et al. “Hierarchical dirichlet processes”. In: *Journal of the american statistical association* 101.476 (2006), pp. 1566–1581.

[16] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. ” O’Reilly Media, Inc.”, 2009.

[17] Matthew Honnibal and Ines Montani. “spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing”. To appear. 2017.

[18] Radim Rehurek and Petr Sojka. “Gensim–python framework for vector space modelling”. In: *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic* 3.2 (2011).

[19] Guido Van Rossum. *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020.

[20] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[21] Victor Sanh et al. “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter”. In: *arXiv preprint arXiv:1910.01108* (2019).

[22] James MacQueen et al. “Some methods for classification and analysis of multivariate observations”. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297.

[23] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[24] Ville Satopaa et al. “Finding a” kneedle” in a haystack: Detecting knee points in system behavior”. In: *2011 31st international conference on distributed computing systems workshops*. IEEE. 2011, pp. 166–171.

[25] Aditya Grover and Jure Leskovec. “node2vec: Scalable feature learning for networks”. In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016, pp. 855–864.

[26] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. https://christophm.github.io/interpretable-ml-book/. 2019.

[27] Charu C. Aggarwal. *Recommender Systems: The Textbook*. 1st. Springer Publishing Company, Incorporated, 2016. ISBN: 3319296574.

[28] Francesco Ricci et al. *Recommender Systems Handbook*. 1st. Berlin, Heidelberg: Springer-Verlag, 2010. ISBN: 0387858199.

[29] Google Developer. *Recommendation Systems*. URL: https://developers.google.com/machine-learning/recommendation.

[30] Xiangnan He et al. “Neural collaborative filtering”. In: *Proceedings of the 26th international conference on world wide web*. 2017, pp. 173–182.

[31] Donghyun Kim et al. “Convolutional matrix factorization for document context-aware recommendation”. In: *Proceedings of the 10th ACM conference on recommender systems*. 2016, pp. 233–240.

[32] Jeffrey Pennington, Richard Socher, and Christopher D Manning. “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[33] Clayton Hutto and Eric Gilbert. “Vader: A parsimonious rule-based model for sentiment analysis of social media text”. In: *Proceedings of the International AAAI Conference on Web and Social Media*. Vol. 8. 1. 2014.

[34] Jaesung Huh et al. “Augmentation adversarial training for self-supervised speaker recognition”. In: *arXiv preprint arXiv:2007.12085* (2020).

[35] Bishan Yang et al. “Embedding entities and relations for learning and inference in knowledge bases”. In: *arXiv preprint arXiv:1412.6575* (2014).

[36] Greg Benton et al. “Trajectory Based Podcast Recommendation”. In: *arXiv preprint arXiv:2009.03859* (2020).

[37] Ignacio Fernández-Tobías et al. “Addressing the user cold start with cross-domain collaborative filtering: exploiting item metadata in matrix factorization”. In: *User modeling and user-adapted interaction* 29.2 (2019), pp. 443–486.

[38] Kyung Soo Kim, Doo Soo Chang, and Yong Suk Choi. “Boosting Memory-Based Collaborative Filtering Using Content-Metadata”. In: *Symmetry* 11.4 (2019), p. 561.

[39] Shoujin Wang et al. “A survey on session-based recommender systems”. In: *arXiv preprint arXiv:1902.04864* (2019).

[40] Fei Yu et al. “Network-based recommendation algorithms: A review”. In: *Physica A: Statistical Mechanics and its Applications* 452 (2016), pp. 192–208.

[41] Andrew Zimdars, David Maxwell Chickering, and Christopher Meek. “Using temporal data for making recommendations”. In: *arXiv preprint arXiv:1301.2320* (2013).

[42] Guy Shani, David Heckerman, and Ronen I. Brafman. “An MDP-Based Recommender System”. In: *Journal of Machine Learning Research* 6.43 (2005), pp. 1265–1295. URL: http://jmlr.org/papers/v6/shani05a.html.

[43] Feng Liu et al. “Deep reinforcement learning based recommendation with explicit user-item interactions modeling”. In: *arXiv preprint arXiv:1810.12027* (2018).

[44] Nima Taghipour and Ahmad Kardan. “A hybrid web recommender system based on q-learning”. In: *Proceedings of the 2008 ACM symposium on Applied computing*. 2008, pp. 1164–1168.

[45] Xinxi Wang et al. “Exploration in interactive personalized music recommendation: a reinforcement learning approach”. In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 11.1 (2014), pp. 1–22.

turing.ac.uk
@turinginst