# Beyond Metadata for BBC iPlayer: an autoencoder-driven approach for embeddings generation in content similarity recommendations

Simone Spaccarotella

4th November 2024

# Contents

# Listings



# List of Figures

# 1 Introduction and background

I am a software engineer at the BBC and team lead for the Sounds web team. I also trained as a data scientist and worked in attachment to the iPlayer Recommendation team.

I built a machine learning model pipeline that generates content-to-content (C2C) similarity recommendations of video-on-demand (VOD) for the "More Like This" section on BBC iPlayer [3]. This project is relevant to me because I have been crossing paths with the world of recommendations multiple times during my career at the BBC, which sparked my interest. I had a tangent encounter in 2015 while working for a team that built an initial recommender for BBC News and an API to provide recommendations using third-party engines. I also produced and presented a talk for a Hack Day. The talk was called "Recommendation Assumptions" [17], and it was about types of recommendations and contextual external factors affecting them. Until now, when I could finally put my knowledge into practice working on an actual project on real data.

The BBC is a well-known British broadcaster that is constantly evolving to remain relevant to its audience. Its mission is to inform, educate, and entertain, and it operates within the boundaries set by the Royal Charter [8]. The current media landscape requires the BBC to deliver digital-first content relevant to the audience. This transformation involves investments in data and personalised services, not to mention an inevitable revolution in generative machine learning modelling that is keeping everyone busy.

# 2 Outline of the issue, opportunity and the business problem to be solved

The BBC produces and stores vast amounts of data for its content, surfaced by countless services and APIs. One of the top priorities for the BBC is to increase the usage across the business of **Passport** [9], an internal BBC system that provides a rich set of metadata annotations for multimodal content (audio, video and text). The usage in production is low, and its BBC-wide adoption would make access to metadata consistent, remove duplications, and reduce effort and costs.

In addition, the current content similarity calculation uses a suboptimal approach, which limits the quality of the recommendations. The similarity score is directly proportional to the number of values in common between any pairs of items on a per-feature basis. However, the commonality is calculated with exact string equality, ignoring any relationship between different categorical values expressing a similar concept (e.g. "comedy" and "stand-up comedy"). Moreover, the limited number and type of tags cannot adequately describe the content. At the same time, the skewness of the data distribution and a lack of pre-processing bias the recommendations towards the high-frequency values. Lastly, each similarity score is multiplied by a hardcoded weight that modulates the importance of a feature, but it does not solve the polarising effect of a skewed distribution. Unfortunately, because these are hyperparameters and not learned weights, the model cannot improve the performance by minimising them against a cost function.

This project could help the BBC enhance the quality of recommendations while reducing costs and accelerating innovation, thus increasing the licence fee value for money. Its contribution could:

- **Improve the quality of the content similarity recommendations**. This solution ingested a rich set of metadata that better described the content and applied a novel technique that improved the descriptive power of the transformed metadata, improving the similarity calculation quality.
- **Build a foundational item-embeddings generator for upstream recommenders**. This project provided an immediate solution for non-personalised C2C recommendations that solely rely on content metadata. It also provided a foundational approach for embedding generation for upstream personalised recommenders, which could improve the

quality of the personalised recommendation and increase user engagement.

- **Reduce costs by building a general solution for the wider BBC**. The pipeline ingested a dataset of content metadata composed of tags used to annotate all BBC content, making this solution general and reducing duplications and costs.

# 3 Methods and justification

## 3.1 Data pre-processing

I used **one-hot encoding** to transform the categorical features (i.e., the metadata annotations) into a numerical vector. This simple yet effective encoding method is perfect for transforming nominal categoricals because it does not introduce any ranking or arithmetic relationship among the encoded values. The downside of this approach was that it generates high-dimensional sparse arrays, introducing the so-called *curse of dimensionality*, an issue handled by the chosen deep-learning modelling technique.

## 3.2 Modelling and regularisation

I used a **neural network** to compress the high-dimensional one-hot encoded vectors to a lower-dimensional embedding representation. This technique captured non-linearity by learning the underlying latent structure of the data, which was essential to represent the original Passport tags in a geometrical space, exploiting local proximity as a similarity measure.

I used **hyperparameter tuning** to find the best model parameters that minimised the cost function and used the validation set to regularise the model with **early stopping**. This technique monitored the reconstruction loss on an out-of-sample dataset, allowing the model to stop training within a set "patience" threshold after reaching a local minimum on the validation error. Finally, I used the test set to assess the model performance using the best set of parameters.

I used **dropout** to regularise the model further and make it robust to small fluctuations in the input, and **data augmentation**, by including in the training data the episodes that shared the same tags with their parent programme. **Weight decay** and **batch normalisation** were tested during hyperparameter tuning and discarded for poor performance.

## 3.3 Content similarity

I used the **cosine** of the angle $\theta$ between each pair of embeddings to calculate the similarity scores. This metric is insensitive to the magnitude of the vectors, and because high-frequency values tend to have larger magnitudes, it mitigates the impact on recommendations of commonly used tags.

One-hot encoded vectors lack meaningful relations between them. They represent unit vectors bound in the "positive quadrant" of a Cartesian coordinate system for a multidimensional Euclidean space. Because each pair

can only have a finite number of angles, the cosine similarity will also assume a finite number of discrete values between zero and one, causing information loss. Ideally, we would expect the similarity score to be a continuous value bound between `-1` and `1`, and this is only possible if the angle $\theta$ of any vector pair is a number between `0` and `360` (i.e. $0\pi$ and $2\pi$), hence the use of embeddings.

## 3.4 Tools and frameworks

The entire project was written in **Python**. It is the *de facto* programming language for data science and machine learning tasks. Python has an established, diverse, and well-documented ecosystem of external libraries and frameworks that facilitate the job, and it is also the language of choice at the BBC.

I used **Pandas** only for tabular data manipulation to generate and store the one-hot encoded vectors. Unfortunately, using it for exploratory data analysis (EDA) was impossible because the iPlayer catalogue used in development had roughly one year's worth of data, which did not fit in memory, causing Pandas to crash. Therefore, I used **Dask**, a library capable of running out-of-memory and parallel execution for faster processing on single-node machines and distributed computing on multi-node machines while using the familiar Pandas API.

I used **TensorFlow** and **Keras** for modelling to build and train the encoder-decoder neural network architecture and **Keras Tuner** for hyperparameters optimisation. I also used **Scikit-learn** but not for modelling. It provided utility functions for the dataset splitting and the cosine similarity calculation, and I was already familiar with its API.

I used a combination of **Matplotlib** and **Seaborn** for visualisation and **rdflib** to fetch and parse the RDF documents that contained the labels needed to visualise the recommendations for testing purposes. It is worth mentioning the use of **pytest** for unit testing and **black** for PEP 8 code compliance and formatting. Finally, I used **Jupyter Lab** to edit the project, **git** for code versioning, **GitHub** as a remote code repository and for collaboration, and **AWS Sagemaker** to run the pipeline on more capable virtual machines, especially during hyperparameter tuning.

# 4 Scope of the project and Key Performance Indicators

This project's scope was to build an end-to-end machine learning solution that could produce non-personalised content-to-content similarity recommendations using Passport metadata tags as input.

The minimum viable outcome was to produce recommendations comparable to those currently in production, while the desired outcome was to increase user engagement. The integration with Passport would make this solution general and applicable to multimodal BBC content. If adopted by $N$ BBC products with a total cost of $C$, it could generate considerable savings, with an approximate cost reduction by a factor of $\frac{C}{N}$.

Comparability was a qualitative and subjective key performance indicator (KPI) that served as a compass, indicating whether the project was progressing towards the right direction. Technical and non-technical stakeholders with diverse domain knowledge and background were involved. I built a rudimentary tool that visualised the programme's title, image, description and Passport tags, comparing the seed programme with the top-K recommendations. The people involved gave their subjective feedback on their perceived level of similarity of the output, testing edge cases and sensitive recommendations like content recommendations for children's accounts. They also discussed anomalies and unexpected results. Opening the "More Like This" tab on any programme page on BBC iPlayer allowed them to compare the live recommendations with those generated by the pipeline.

The solution needed to be A/B tested in production, with live data, to measure user engagement. Unfortunately, too many moving parts outside my control were required to happen for me to build a production-worthy version to achieve that. Adopting this as a KPI would have delayed the project, increasing the odds of failure. To mitigate this risk, I had to decouple it from the project's success.

I defined a hypothesis that could be tested offline and within my control. The hypothesis stated that long-term user engagement is not just about accuracy. It can be affected by increasing diversity in the recommended content. A diverse set of recommendations generates new and unexpected results, increasing surprise and serendipity, pushing the user away from boredom. This theory was untested but grounded in active research on the topic, such as [13] and [10], which made it less far-fetched. For this reason, I decided to measure diversity offline based on the frequency and distribution of the tags that annotated the recommendations, pending future A/B testing to validate the hypothesis.

# 5 Data selection, collection and pre-processing

BBC News and Sport articles, iPlayer videos, and Sounds audios are annotated with Passport tags. These tags describe any BBC content and can be used for retrieval (search) and filtering (recommendations). They can be applied manually by an editorial team with domain knowledge or semi-automatically by machine learning algorithms with human supervision.

Passport tags are distributed across the BBC via the universal content exposure and delivery (UCED) system. This self-service delivery platform exposes data as a document stream for products to integrate. It provides different types of consumers, such as REST API, AWS S3 bucket, etc. Passport documents are JSON objects that contain a property called "`taggings`", an array of objects representing the metadata annotations. Two properties describe these objects: "`predicate`" and "`value`". They represent a tag's name and value are expressed as URL-formatted strings, except for dates [figure 1].


Figure 1: An example of Passport file

The predicate is a class of the BBC Ontologies [2] and describes the annotation. The value can be a date or a BBC Things entity [5, 6] defined as an RDF document [21, 19], described by other BBC Ontology classes and accessible in Turtle format [20] via the BBC Things API [7]. These entities are linked to each other and other external resources and form a graph data structure.

I decided not to integrate with UCED during development but to use batches of Passport files, manually collected and stored in a local folder. This trade-off allowed me to train the model with live data while reducing costs. In addition, because I had to create resources on two AWS accounts, I did not want to pass the burden of maintenance to the team that owned them without having tested the feasibility of the solution first.

Content metadata does not constitute personal data and, therefore, is not subject to the UK GDPR [12]. Nonetheless, this data is encrypted at rest and in transit by default. For this reason, no further actions were required during storage and processing.

I chose to use Passport because it provides a set of tags shared across all BBC content, making this a general solution that reduces duplications and costs. Passport offers a flexible and rich set of tags to describe any type of content. The eight annotations I selected for training were: `about`, `genre`, `format`, `contributor`, `motivation`, `editorialTone`, `narrativeTheme` and `relevantTo` (see figures 13, 14, 15, 16, 17, 18, 19 20 and 21 for example values).

A programme of the BBC iPlayer catalogue is mapped internally by a hierarchy of items defined by an ID called *PID*. An item can be of type *episode*, *series*, or *brand*. The *episode* is the leaf of this tree-like structure, representing the playable content like an episode of a series, a live show, or a one-off, like a movie or documentary. In contrast, *series* and *brand* types are considered "containers" and appear at the upper levels of the tree. Their children can be other containers or *episode* items [4, 18], [figure 2] and [figures 27, 28, 29 of the appendix].

| pid | parent_pid | tleo_pid | parent_type | tleo_type |
|---|---|---|---|---|
| **m0018cgz** | m0017wzn | b007qgm3 | series | brand |
| **m001c98h** | m001c1gf | m001c1gg | series | brand |
| **m001vh3q** | b07gfjzn | b006pfr1 | series | brand |
| **b05y0mzt** | b052vhsb | b00sp0l8 | series | brand |
| **m001n4jx** | b07jltpv | b00cpbm9 | series | brand |
| **m001g8cn** | NaN | m001g8cn | NaN | episode |
| **b0b3fly3** | b09s2rf1 | b09s2qb6 | series | brand |
| **b018nvwc** | b00x85x3 | b01pw0b2 | series | brand |
| **b037m1zp** | b0274dph | b00sp0l8 | series | brand |
| **m0012z4t** | m0010m3d | m000mh7n | series | brand |
| **p0b5p2l6** | p0b5635c | p0b562cl | series | brand |
| **m001qn6f** | b006nldz | b006nldz | brand | brand |
| **m001d3h9** | NaN | m001d3h9 | NaN | episode |
| **m001kqyc** | b0070x47 | b0070x47 | brand | brand |
| **b04nyk9y** | b04lx10l | b006t1p7 | series | brand |

Figure 2: Internal programme hierarchy structure

The dataset used for training contained the encoded tags of the episode and container items composing the `5934` programmes that were available

in the iPlayer catalogue from `13 June 2023` to `15 April 2024`. Of the `83098` items, `81311` were annotated with Passport tags stored in JSON files, amounting to a coverage of `97.85%`. Because each item of this tree-like structure represents a part of the programme (a single episode, a season, or the programme itself) and shares the Passport tags by inheritance, some of the dataset rows had similar or duplicate encoded tags belonging to the episodes of the same programme. This information redundancy worked as a data augmentation training technique to improve the model's performance, considering the small size of the catalogue compared to the `8982` one-hot encoded features.

The pipeline's pre-processing stage loaded these files [figure 1] into a dictionary data structure [code 1]. The dictionary's key was the *PID* and the value was another dictionary describing the annotations [figure 3]. A pro-

| | about | format | contributor | genre | motivation | editorialTone | narrativeTheme | relevantTo |
|---|---|---|---|---|---|---|---|---|
| **b05sxyhw** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **m001rrx4** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **b06wjqf9** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **m000rppl** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **b09sv1f5** | [3b32791c-d26f-4d43-992e-f0538e2dd1a2, 231d9a6... | [6023be6b-1ab2-4572-8129-64c24a262abf] | NaN | NaN | NaN | NaN | NaN | NaN |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **m001bvvg** | NaN | [c595e555-ebb4-4d46-b150-a6bb3ca4a25e] | NaN | NaN | NaN | NaN | NaN | NaN |
| **b00wbkw5** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **p0fjh78f** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **m001qgg4** | NaN | [c595e555-ebb4-4d46-b150-a6bb3ca4a25e] | NaN | NaN | NaN | NaN | NaN | NaN |
| **m001xjjw** | NaN | [fa8834af-f256-4e97-bf3f-38176114c237] | [b106d6a9-4d8a-4dad-b259-1b52bc3d11a1] | [83d5b106-f331-4f5a-bf08-46ff4ebd6032] | NaN | NaN | NaN | NaN |

81311 rows × 8 columns

Figure 3: A DataFrame visualisation of the dictionary data structure

gramme can be tagged with the same predicate multiple times [figure 4] if it has different values, while the same value (e.g. "Music") can be used by various predicates (e.g. `about` or `genre`). The pipeline then created a

```
about             [14745d1f-885d-4b9f-b28a-24540e7beb15, 65db201...
format                        [6023be6b-1ab2-4572-8129-64c24a262abf]
contributor                                                      NaN
genre             [26a553a2-8995-4117-8570-c23499eb5c52, a43fa03...
motivation                    [b1a660eb-ef14-4645-b4cd-d7bd939ce443]
editorialTone     [d6f8e5d3-cd6f-46d6-9aa9-a5049735893a, 8eda393...
narrativeTheme                                                   NaN
relevantTo                    [e7c12623-1b0f-4d8c-a624-af34744b7cf4]
Name: m000fvf5, dtype: object
```

Figure 4: A sample programme with tagging

Pandas *Dataframe* pre-populated with zeros, where the rows represented the programmes, and the columns represented the tags. I used a MultiIndex [16] for the columns because it needed to keep track of the duplicate values (2nd-level index) across the predicates (1st-level index) to access the cells to populate them with the value "`1`" if the programme was annotated with the corresponding tag, generating one-hot encoded array of size `8982` [figure 5] [code 2].

| | 2225f74b-510c-4602-9589-250f98ecaf24 | cf825117-9051-4230-9086-183c51db99f4 | ec5e3279-c5a2-4f65-a326-124cbfdf3619 | b248d912-9be9-463d-ac83-dab354fc1789 | 02b76e94-a117-4ffc-b754-914c44d91152 | c781574c-0676-4a41-919a-56d7c9adf75d | a8ce7b94-0526-40f1-a1c0-a0d6f042131f | a9770520-b9fb-4633-8fa0-585b33b6c7c1 | 41aa7fbf-f1ff-4769-9fbb-ff6e5f89da94 | ed1dd219-3877-4b51-aa84-04afc21d59d9 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| b05sxyhw | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| m001rrx4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| b06wjqf9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| m000rppl | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| b09sv1f5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| m001bvvg | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| b00wbkw5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| p0fjh78f | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| m001qgg4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| m001xjjw | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |

81311 rows × 8982 columns

Figure 5: The one-hot encoded dataset

One-hot encoding is positional and does not care about the actual tag labels, so I decided to extrapolate the URL identifiers [figure 6] and use them as tag values, speeding up the data loading stage because it did not need to fetch the labels from the BBC Ontology.

```
['14745d1f-885d-4b9f-b28a-24540e7beb15',
 '65db2010-982d-4442-adca-a7af6dd6d55f',
 '074c2f99-fe40-496b-a072-5a0750211999',
 '12e69b92-a7ba-4463-84e0-be107b9805d0',
 '49b34f66-9ef6-4476-9fec-bf82f0b944f1',
 '0fe34065-ad3d-477e-b97d-1e6ddd669ac4',
 '15f1bcf6-b6ab-48e8-b708-efed41e43d31',
 '28022138-bb15-4d92-afb2-a790aa4e1b71']
```

Figure 6: A list of hashed values for a given tag

A source of bias in the dataset was the `mentions` tag. This annotation type is automatically generated by an algorithm that extracts terms deemed important, appearing in the text of an article or the transcript of an audio/video content. If something is mentioned, it does not necessarily describe the content because of the intrinsic ambiguities of natural languages. Figure of speech devices, such as metaphors, analogies, allegories, and others, alter the meaning of a sentence for stylistic effect and can misrepresent the main topic. For example, if the phrase "being over the moon" is mentioned

by someone delighted about something unrelated to the topic of "space" and "universe", extracting the term "moon" as a descriptor could mislead the representation. To mitigate this source of bias, I dropped `mentions` in favour of `about`, a tag that describes the main topics of the programme, annotated by a team of editorials with domain knowledge, trained in unconscious bias management and how to use relevant tags for a given programme.

Generating embeddings of one-hot encoded vectors with the same size used in training but with unseen tags leads to unpredictable errors. The encoding is positional, and the combination of `1` and `0` learned by the model belongs to the tags seen during training. So, I decided to drop the new tags and encode only the ones the model was trained on while padding the rest with zeros, pending retraining to capture the new information. Some programmes did not have any annotations, producing an encoding of all zeros, so I dropped them and included only the ones with at least one annotation.

# 6 Survey of potential alternatives

**Clustering** the one-hot encoded features to group similar items was not viable. Recommenders return ranked lists, while clustering would have returned an unordered list of items belonging to the same cluster as the one considered for similarity. Furthermore, the number of clusters was unknown. Even using a density-based technique to auto-discover them would not have helped because similarity ranking requires every pair of items to have a score, and grouping does not map with this concept. Clustering was a coarse-grained discretisation of similarity, and I needed a more granular approach where every item could be compared with everything else.

The intuition was to map the concept of similarity with a geometric interpretation and calculate the **pairwise distance** between the vectors representing the items, given a metric and a vector space relevant to the similarity measure. So, I discarded clustering as a candidate option.

One-hot encoding does not use spatial proximity information to transform the categorical features into ones and zeros. It is a transformation process that pivots the unique values of each original feature to be the new variables of the transformed vector. If we project these raw vectors in a multidimensional space, we cannot use their relative position to each other as a similarity measure. Moreover, the high dimensionality of the vectors would have increased the computational complexity.

To calculate the pairwise distance efficiently and produce a meaningful representation of similarity, I had to transform the vectors in a denser and lower dimensional space, a manifold embedded into the original high-dimensional ambient space.

I considered dimensionality reduction techniques such as principal component analysis (PCA), independent component analysis (ICA) or linear discriminant analysis (LDA), but there was a problem with them too. Their job is to find a linear projection of the data, but this is a strong assumption that misses important non-linear structures. I did not use **PCA** (which I was more familiar with), but I did not discard the idea of using dimensionality reduction. I just needed a non-linear approach and turned my attention to **manifold learning**.

I could have chosen an entirely different encoding technique to generate lower-dimensional vectors and avoid the curse of the dimensionality problem from the beginning, so I also investigated **feature hashing**. A hash function is a non-invertible function that can map data of arbitrary size to fixed-size values typically used for constant lookups. Hashing can also be used as a feature transformation technique for tabular data. It can generate smaller vectors than one-hot encoding but does not encode any concept of similarity

either. I did not adopt it because, in comparison, it has more hyperparameters, which increases its complexity. In addition, although unlikely, it introduces the collision problem, where two distinct inputs can be mapped to the same index in the same target domain. This issue could be mitigated by choosing the latest and most robust algorithm to reduce the likelihood of that happening. However, the trade-off was too computationally expensive for pre-processing. The reality was that I needed a pre-processing technique to transform categorical features into a high-dimensional numerical vector to artificially inflate the encoding dimension for the modelling technique I had in mind since the beginning to produce quality embeddings.

# 7 Implementation and performance metrics

I trained an **autoencoder** [1, 15] to learn the Passport tags' latent features and reduce the encoded vectors' size. The autoencoder is an encoder-decoder neural network, a self-supervised model capable of capturing non-linearity from the data. I used the "undercomplete" variant, which constrains the number of nodes in the hidden layers, creating a "bottleneck" of information flow through the network. This architecture is a form of *regularisation* that forces the model to learn latent attributes from the input while reconstructing it with minimal loss. Ultimately, it helps prevent the model from overfitting the training dataset by indexing it like a caching layer. The autoencoder has a symmetrical architecture composed by the *encoder* and the *decoder* sub-networks, with a bottleneck layer in the middle also known as *code* [figure 7 and 5].

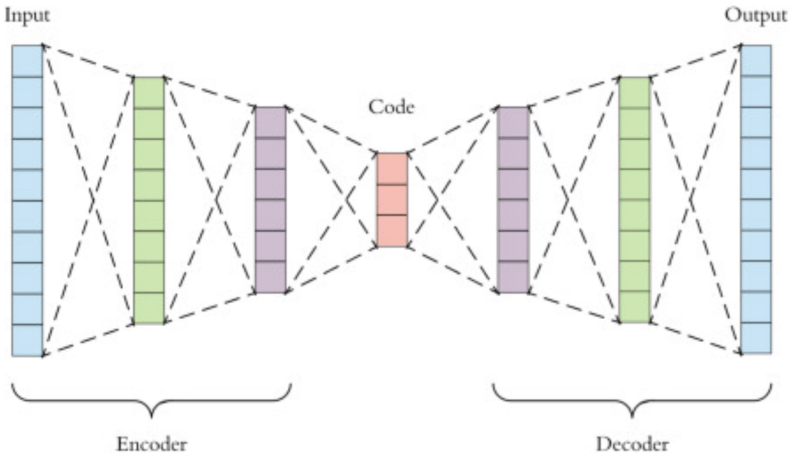



Figure 7: Undercomplete autoencoder architecture

I extracted the trained *encoder* segment of the network to compress the one-hot encoded high-dimensional sparse array into a lower-dimensional and denser representation called *embeddings* [11] [code 8]. This technique solved the curse of dimensionality and the data sparsity problems and improved the calculation complexity and the quality of the recommendations at inference time [code 9].

## 7.1 Hyperparameters

Some of the hyperparameters were decided based on the nature of the problem. The input data was a tensor of zeros and ones, and the network's main objective was to reconstruct the output with minimal loss. I used the *Sigmoid* activation function for the output layer and *binary cross-entropy* as a loss function to allow the network to minimise the reconstruction error, pushing the values of the output between zero and one. The minimisation of the reconstruction error was the reason for using *binary cross-entropy* as the performance metric. I did not use the *SoftMax* activation function for the output layer because every node needed to be able to assume those values. I did not use any residual-based loss function either because the penalty for wrong predictions (i.e., when the X-axis value is close to zero) is not as significant as for the binary cross-entropy loss. Because the slope of the derivative in this region is not steep enough, it also affects the step size during back-propagation [figure 8]. The negative log loss was the best option because it heavily penalised significant differences between $y$ and $\hat{y}$, with a natural logarithmic progression.

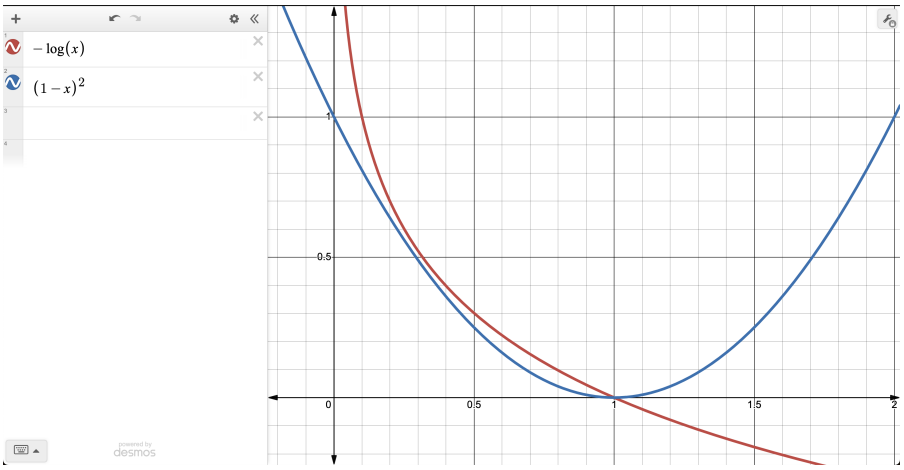



Figure 8: Comparison between Binary Cross-Entropy and Square Residuals loss functions

I used the *Rectified Linear Unit (ReLU)* as an activation function for the hidden layers while testing other variants like *Leaky ReLU (LReLU)* and *Parametric ReLU (PReLU)*, but without any relevant improvements in performance. Some other hyperparameters were:

- **optimizer:** `Adam`

- **number_of_epochs:** `100`
- **batch_size:** `300`
- **dropout_rate:** `0.2`
- **early_stopping_patience:** `10`
- **early_stopping_monitoring:** `binary cross-entropy` on the validation set
- **data_split_ratio:** `80% training, 10% validation, 10% test`

I used a "Bandit-based" approach called *Hyperband* [14] to optimise the remaining hyperparameters [codes 4, 5]. It improves upon *Random Search* by running fewer epochs on the randomly sampled set of parameters and moving on to the next stage by only testing the best-performing ones, returning a ranked list of the best hyperparameter sets. The hyperparameters considered by Hyperband were:

- the number of hidden layers
- the embedding size
- the learning rate
- whether to use dropout
- whether to use batch normalisation

The number of hidden layers was a positive odd integer $N > 0$. The network had the encoder and decoder with $N - 1$ layers and the bottleneck layer in the middle. If the value were 1, the network would only have the bottleneck.

The number of nodes per layer was a positive integer $N > 0$. Unfortunately, the number of combinations was too high to be meaningfully optimised. To reduce the complexity, I decided to limit the values of this hyperparameter to a progression of integer divisions by 2, starting from the number of nodes in the input layer. Each hidden layer in the encoder could have half the number of nodes compared to the previous one and double the amount of the successive one (and *vice versa* for the decoder), except for the layers adjacent to the input and output. In that case, they could assume any progression value depending on the number of hidden layers and embedding size.

For example, because I had 8982 nodes in input, the progression of layer dimensions was `[4491, 2245, 1122, 561, 280, 140, ...]`. The embedding size was also bound to assume one of these values, which would determine the maximum number of hidden layers allowed. Therefore, a network with `5` hidden layers and an embedding size of `280` would have had the following configuration: `8982 -> 1122 -> 561 -> 280 -> 561 -> 1122 -> 8982`. While a network with `3` hidden layers but an embedding size of `2245` would have had the following configuration: `8982 -> 4491 -> 2245 -> 4491 -> 8982`, representing the maximum extension of the progression.

## 7.2 Training and validation

To improve and assess the ability of the model to *generalise* on unseen data, I randomly shuffled the dataset and split it into three chunks: training, validation and test [code 3]. The training phase took `20h 55m 11s` on Sagemaker to tune the hyperparameters, returning the following best values:

- **hidden_layers:** `1`
- **embeddings_size:** `561`
- **batch_norm:** `False`
- **dropout:** `True`
- **learning_rate:** `0.01`

The trained model had `8982` nodes for the input and output layers - which corresponded to the size of the one-hot encoded array - and `561` nodes for the bottleneck layer, with a total of `10,087,347 (38.48 MB)` parameters [figure 9]

**Model: "Autoencoder"**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| code (Dense) | (None, 561) | 5,039,463 |
| dropout (Dropout) | (None, 561) | 0 |
| output (Dense) | (None, 8982) | 5,047,884 |

**Total params:** 10,087,347 (38.48 MB)

**Trainable params:** 10,087,347 (38.48 MB)

**Non-trainable params:** 0 (0.00 B)

Figure 9: Autoencoder model summary

Training completed with a reconstruction error of `0.0002478554961271584` on the validation set [figure 10] [codes 6, 7].

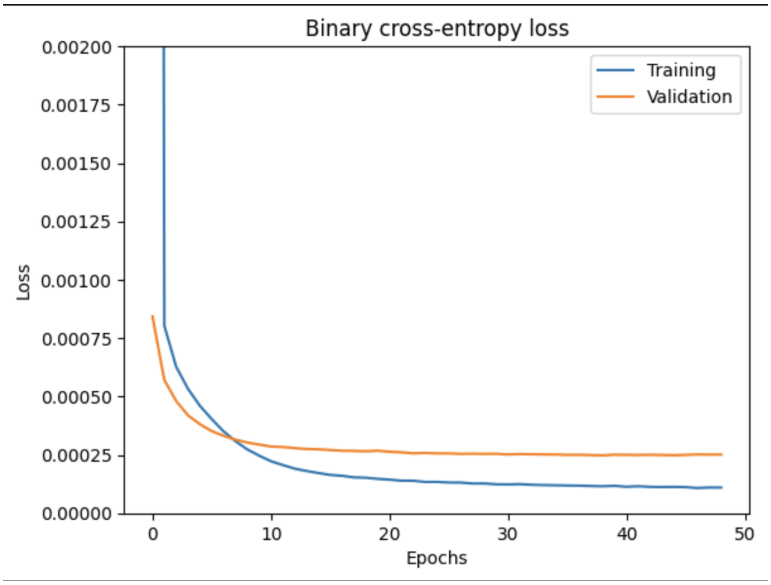



Figure 10: Binary cross-entropy loss on the training and validation sets

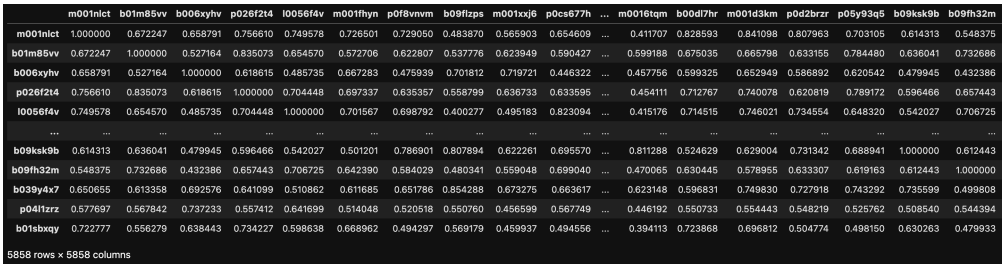
## 7.3 Strengths and Weaknesses

The strength of this approach was the efficient use of space and the fast inference time. The space scaled linearly with respect to the input size to store the embeddings [figure 11] and quadratically to store the similarity scores [figure 12].

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 551 | 552 | 553 | 554 | 555 | 556 | 557 | 558 | 559 | 560 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **m001nkfq** | 0.430479 | 0.267473 | 0.476859 | 0.528339 | 0.282628 | 0.038342 | 0.586313 | 0.396646 | 0.045122 | 0.372976 | ... | 0.330867 | 0.481218 | 0.385905 | 0.108823 | 0.492268 | 0.335194 | 0.710101 | 0.486175 | 0.529714 | 0.000000 |
| **m001nlct** | 0.793537 | 0.056660 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.873273 | 0.000000 | 0.000000 | ... | 0.000000 | 0.031108 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **m001pb9q** | 0.436232 | 0.148591 | 0.472391 | 0.291805 | 0.521484 | 0.195137 | 0.472602 | 0.280848 | 1.028414 | 0.322516 | ... | 0.265798 | 0.263675 | 0.422720 | 0.000000 | 0.558023 | 0.392738 | 0.298900 | 0.277357 | 0.386545 | 0.000000 |
| **b08nz382** | 0.133345 | 0.263294 | 0.323048 | 0.256369 | 0.149547 | 0.369113 | 0.283260 | 0.124693 | 0.176905 | 0.307855 | ... | 0.391727 | 0.271868 | 0.207543 | 0.160561 | 0.354360 | 0.257897 | 0.594704 | 0.151107 | 0.358588 | 0.613461 |
| **m001mx54** | 0.268269 | 0.402586 | 0.245007 | 0.127917 | 0.215340 | 0.275689 | 0.112461 | 0.411956 | 0.879633 | 0.253204 | ... | 0.134891 | 0.210231 | 0.098260 | 0.314244 | 0.197673 | 0.308381 | 0.186477 | 0.289726 | 0.218524 | 0.249002 |

5 rows × 561 columns

Figure 11: Embeddings dataset



| | m001nlct | b01m85vv | b006xyhv | p026f2t4 | l0056f4v | m001fhyn | p0f8vnvm | b09flzps | m001xxj6 | p0cs677h | ... | m0016tqm | b00dl7hr | m001d3km | p0d2brzr | p05y93q5 | b09ksk9b | b09fh32m |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **m001nlct** | 1.000000 | 0.672247 | 0.658791 | 0.756610 | 0.749578 | 0.726501 | 0.729050 | 0.483870 | 0.565903 | 0.654609 | ... | 0.411707 | 0.828593 | 0.841098 | 0.807963 | 0.703105 | 0.614313 | 0.548375 |
| **b01m85vv** | 0.672247 | 1.000000 | 0.527164 | 0.835073 | 0.654570 | 0.572706 | 0.622807 | 0.537776 | 0.623949 | 0.590427 | ... | 0.599188 | 0.675035 | 0.665798 | 0.633155 | 0.784480 | 0.636041 | 0.732686 |
| **b006xyhv** | 0.658791 | 0.527164 | 1.000000 | 0.618615 | 0.485735 | 0.667283 | 0.475939 | 0.701812 | 0.719721 | 0.446322 | ... | 0.457756 | 0.599325 | 0.652949 | 0.586892 | 0.620542 | 0.479945 | 0.432386 |
| **p026f2t4** | 0.756610 | 0.835073 | 0.618615 | 1.000000 | 0.704448 | 0.697337 | 0.635357 | 0.558799 | 0.636733 | 0.633595 | ... | 0.454111 | 0.712767 | 0.740078 | 0.620819 | 0.789172 | 0.596466 | 0.657443 |
| **l0056f4v** | 0.749578 | 0.654570 | 0.485735 | 0.704448 | 1.000000 | 0.701567 | 0.698792 | 0.400277 | 0.495183 | 0.823094 | ... | 0.415176 | 0.714515 | 0.746021 | 0.734554 | 0.648320 | 0.542027 | 0.706725 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **b09ksk9b** | 0.614313 | 0.636041 | 0.479945 | 0.596466 | 0.542027 | 0.501201 | 0.786901 | 0.807894 | 0.622261 | 0.695570 | ... | 0.811288 | 0.524629 | 0.629004 | 0.731342 | 0.688941 | 1.000000 | 0.612443 |
| **b09fh32m** | 0.548375 | 0.732686 | 0.432386 | 0.657443 | 0.706725 | 0.642390 | 0.584029 | 0.480341 | 0.559048 | 0.699040 | ... | 0.470065 | 0.630445 | 0.578955 | 0.633307 | 0.619163 | 0.612443 | 1.000000 |
| **b039y4x7** | 0.650655 | 0.613358 | 0.692576 | 0.641099 | 0.510862 | 0.611685 | 0.651786 | 0.854288 | 0.673275 | 0.663617 | ... | 0.623148 | 0.596831 | 0.749830 | 0.727918 | 0.743292 | 0.735599 | 0.499808 |
| **p04l1zrz** | 0.577697 | 0.567842 | 0.737233 | 0.557412 | 0.641699 | 0.514048 | 0.520518 | 0.550760 | 0.456599 | 0.567749 | ... | 0.446192 | 0.550733 | 0.554443 | 0.548219 | 0.525762 | 0.508540 | 0.544394 |
| **b01sbxqy** | 0.722777 | 0.556279 | 0.638443 | 0.734227 | 0.598638 | 0.668962 | 0.494297 | 0.569179 | 0.459937 | 0.494556 | ... | 0.394113 | 0.723868 | 0.696812 | 0.504774 | 0.498150 | 0.630263 | 0.479933 |

5858 rows × 5858 columns

Figure 12: Similarity scores dataset

Hyperparameter tuning took quite some time, but this drawback was compensated by the fact that training was performed offline while the recommendations were cached and served instantly. In addition, retraining could be scheduled to cover the new programmes added to the iPlayer catalogue and the unavailable ones being removed [figures 23, 24].

The main weakness was the model's interpretability. The autoencoder is a black box by definition, and using embeddings to calculate the similarity just worsened the problem. It was unlikely to interpret which of the tags influenced the ranking in the top-K recommendation by untangling the weights and biases of the neural network, so I analysed the composition and distribution of the tags of the recommended programs as an alternative.

# 8 Discussion and conclusions

## 8.1 Results

This general solution works with any content that uses Passport tags and could provide recommendations for multiple BBC products. Adopting it would reduce effort, duplication of code and data, and, consequently, costs. I shared the findings with the stakeholders, explaining the main benefits and showing the results using the visualisation tool I built. I presented it once to the data scientists and engineers of the iPlayer recommendations team I worked with and another time to the team in charge of the non-personalised recommendations for the entire BBC. The feedback was positive in both cases, and we discussed how to move forward with this project, including the possibility of an A/B test.

## 8.2 Summary of findings and recommendations

The results were perfectly aligned with the initial objectives and measure of success set at the beginning of the project. I recommended building an initial minimum viable product (MVP) consisting of a Sagemaker pipeline built on the AWS development account that ingested batch Passport tags. This recommendation would allow us to break down the engineering effort and spot any blockers/challenges that must be addressed as early as possible to correct them or reconsider some assumptions ahead of the production build.

We would need to build two pipelines, one for training and one for inference, to generate the embeddings and the similarity scores. The embeddings and the similarities score need to be cached to improve performance. The second stage of this approach would require integrating UCED to fetch real-time data automatically.

If this solution is viable and passes the A/B test, it could also be employed to generate embeddings for other personalised recommenders that use item metadata in conjunction with user interactions and contextual data such as day and time of interaction, location and device used.

The project could be further expanded by exploiting the graph nature of the data using a graph neural network (GNN) and, in particular, a graph autoencoder (GAE) to learn a meaningful representation of the graph data, capturing the topological structure and the node content. This extension could improve upon the current autoencoder, which flattens the graph structure in a list of tags and relies on the positional encoding of these tags to generate the embeddings. This effort will require further research and pro-

totyping.

## 8.3 Implications

The project presented a unique opportunity for me to work on an end-to-end machine learning pipeline from data preprocessing to inference, practising my technical skills, building a real neural network, and learning about embedding techniques and content-based recommendation systems. The positive feedback from stakeholders has reinforced my professional confidence and provided invaluable experience in presenting data-driven solutions to a business audience.

For my colleagues and the team, this project has established a replicable framework for C2C similarity recommendations that can be adapted to other BBC products. The solution's modularity enables flexibility in extending it to multimodal content, enhancing the potential for collaborative developments across departments. This adaptability can promote knowledge sharing and foster a data-centric approach to problem-solving in the broader team, as members can leverage this solution to address similar business problems and build upon it

The project presents a scalable solution for stakeholders and the business to reduce data redundancies, decrease maintenance overhead, and potentially reduce costs associated with content recommendation systems. The approach provided a standardised method for generating "More Like This" suggestions, potentially improving user engagement on non-personalised content, with a consistent experience across the BBC portfolio. Moreover, the project's adaptability encourages strategic, data-driven content management across the organisation, supporting future initiatives with robust foundations for content similarity and recommendation. Overall, this project not only aligns with the business goals of optimising resource allocation but also empowers the organisation with a sustainable, scalable recommendation solution for future developments.

## 8.4 Caveats and limitations

Data drift can cause a significant decrease in performance [figure 25], requiring a model re-training. When new programmes are added to the iPlayer catalogue [figure 23], their feature vectors must be encoded, the encoded vectors must be transformed into embeddings, and the new cosine similarities must be re-calculated. Also, both the encodings and the embeddings need to be cached. When the new programmes do not share some or all of the

tags, the encoding phase produces vectors with some or all zeros, indicative of a loss of information.

# 9 Appendices

## 9.1 Code and documentation

### 9.1.1 Data loading and pre-processing

```
import os
import json
import pandas as pd

def get(something, fromTag):
  '''
    Returns the value of the predicate or the tag, extrapolated from
    the URI-formatted string.

    Predicate example: http://www.bbc.co.uk/ontologies/creativework/genre
    Value example: http://www.bbc.co.uk/things/1c3b60a9-14eb-484b-a750-9f5b1aeaac31#id
  '''
    return fromTag.get(something, '').split('/')[-1].split('#')[0]

def should_skip(predicate):
  '''
    Returns True if the Passport tag is not in the allow list
  '''

  # Allow list of predicates to include in the encoding
  return True if predicate not in [
    'about',
    'format',
    'contributor',
    'genre',
    'motivation',
    'editorialTone',
    'narrativeTheme',
    'relevantTo'
  ] else False

tags_dict = dict({})
pids_dict = dict({})
pids_list = []

# Lists the Passport files collected from UCED
uced_files = os.listdir('uced')
```

```
38
39  ###
40  ## Extrapolates the Passport tags per PID
41  ###
42
43  for file_name in uced_files:
44
45    # Gets the PID by splitting "urn:bbc:pips:pid:b05sxyhw.
    json"
46    pid = file_name.split(':')[4].split('.')[0]
47    pids_list.append(pid)
48
49    # Loads the JSON file
50    with open(f'./uced/{file_name}') as json_file:
51        passport = json.loads(json_file.read())
52
53    tags = dict({})
54
55    # For each Passport tag
56    for tag in passport.get('taggings', []):
57      # Extrapolates the tag's name from the URI
58      predicate = get('predicate', tag)
59
60      # Checks the allow list
61      if should_skip(predicate):
62        continue
63
64      # Extrapolates the tag's value from the URI
65      value = get('value', tag)
66
67      # Builds a dictionary of predicates containing a list
    of values related to the current PID
68      if predicate in tags:
69        tags[predicate].append(value)
70      else:
71        tags[predicate] = [value] # list
72
73      # Builds a global dictionary of predicates containing
    all the tag values used for all PIDs.
74      # These tag values may be repeated across predicates.
75      if predicate in tags_dict:
76        tags_dict[predicate].add(value)
77      else:
78        tags_dict[predicate] = {value} # set
79
80    # Builds the dictionary of PIDs, with the associated
    predicate tags
81    pids_dict[pid] = tags
```

Listing 1: Passport tags loading

```
###
## I need to build a dataframe where the rows are indexed by the PIDs, and the columns indexed by the Passport tags.
## To this end, I need to create a multi-index for the columns, where the first level is the tag predicate,
## and the second level the tag value. This will allow duplicate of values across predicates, and access to any
## given cell by speficying the PID, the predicate and the value, in order for the algorithm to set the value "1"
## to signal the PID is tagged.
###

columns = []

for key in tags_dict.keys():
  columns.extend(pd.MultiIndex.from_product([[key], tags_dict.get(key)]))

###
## Creates a zero-filled dataframe of type integer with PIDs as row indexes and the Passport predicates and tags
## as multi-level column index, as specified earlier.
###

df = pd.DataFrame(0, index=pids_list, columns=pd.MultiIndex.from_tuples(columns), dtype='uint8')

###
## Sets 1 for each PID and predicate/tag pair (i.e. One-Hot Encoding)
###

for pid in pids_dict.keys():
  for predicate in pids_dict[pid].keys():
    for tag in pids_dict.get(pid).get(predicate):
      df.at[pid, (predicate, tag)] = 1

###
## Drops the first level of the multi-level column index to flatten the dataframe, now that it is no longer needed.
###

df = df.droplevel(level=0, axis=1)
```

Listing 2: One-hot encoding of the data in a Pandas DataFrame

### 9.1.2 Model training

```
import pandas as pd
from sklearn.model_selection import train_test_split

data = pd.read_parquet('passport_encoding.parquet')

X_train, X_test = train_test_split(data, test_size=0.2)
X_val, X_test = train_test_split(X_test, test_size=0.5)

assert X_train.shape[0] + X_val.shape[0] + X_test.shape[0]
 == data.shape[0]
```

Listing 3: Dataset splitting

```
import tensorflow as tf
from tensorflow import keras
import keras_tuner as kt

# https://www.tensorflow.org/tutorials/keras/keras_tuner

input_size = X_train.shape[1]

# https://keras.io/api/keras_tuner/hyperparameters/
def build_model(hp: kt.HyperParameters):
  # Parameters Set
  hidden_layers = hp.Choice('hidden_layers', [1, 3])
  embeddings_size = hp.Choice('embeddings_size', [70, 140,
 280, 560])
  batch_norm = hp.Boolean('batch_norm')
  dropout = hp.Boolean('dropout')
  learning_rate = hp.Choice('learning_rate', [0.1, 0.01,
 0.001])

  activation = 'relu'
  dropout_rate = 0.2

  model = keras.Sequential()
  model.add(keras.layers.InputLayer(input_shape=(input_size
 ,)))

  if hidden_layers == 1:
    model.add(keras.layers.Dense(
      units=embeddings_size,
      activation=activation
    ))
    if batch_norm:
      model.add(keras.layers.BatchNormalization())
    if dropout:
```

```
32          model.add(keras.layers.Dropout(dropout_rate))
33
34      if hidden_layers == 3:
35        model.add(keras.layers.Dense(
36          units=embeddings_size * 2,
37          activation=activation
38        ))
39        if batch_norm:
40          model.add(keras.layers.BatchNormalization())
41        if dropout:
42          model.add(keras.layers.Dropout(dropout_rate))
43
44        model.add(keras.layers.Dense(
45          units=embeddings_size,
46          activation=activation
47        ))
48        if batch_norm:
49          model.add(keras.layers.BatchNormalization())
50        if dropout:
51          model.add(keras.layers.Dropout(dropout_rate))
52
53        model.add(keras.layers.Dense(
54          units=embeddings_size * 2,
55          activation=activation
56        ))
57        if batch_norm:
58          model.add(keras.layers.BatchNormalization())
59        if dropout:
60          model.add(keras.layers.Dropout(dropout_rate))
61
62      if hidden_layers == 3:
63        model.add(keras.layers.Dense(
64          units=embeddings_size * 4,
65          activation=activation
66        ))
67        if batch_norm:
68          model.add(keras.layers.BatchNormalization())
69        if dropout:
70          model.add(keras.layers.Dropout(dropout_rate))
71
72        model.add(keras.layers.Dense(
73          units=embeddings_size * 2,
74          activation=activation
75        ))
76        if batch_norm:
77          model.add(keras.layers.BatchNormalization())
78        if dropout:
79          model.add(keras.layers.Dropout(dropout_rate))
80
```

```
81        model.add(keras.layers.Dense(
82          units=embeddings_size,
83          activation=activation
84        ))
85        if batch_norm:
86          model.add(keras.layers.BatchNormalization())
87        if dropout:
88          model.add(keras.layers.Dropout(dropout_rate))
89
90        model.add(keras.layers.Dense(
91          units=embeddings_size * 2,
92          activation=activation
93        ))
94        if batch_norm:
95          model.add(keras.layers.BatchNormalization())
96        if dropout:
97          model.add(keras.layers.Dropout(dropout_rate))
98
99        model.add(keras.layers.Dense(
100         units=embeddings_size * 4,
101         activation=activation
102       ))
103       if batch_norm:
104         model.add(keras.layers.BatchNormalization())
105       if dropout:
106         model.add(keras.layers.Dropout(dropout_rate))
107
108       model.add(keras.layers.Dense(input_size, activation='
    sigmoid'))
109
110       model.compile(
111           optimizer=keras.optimizers.Adam(learning_rate=
    learning_rate),
112           loss=keras.losses.BinaryCrossentropy()
113       )
114
115       return model
```

Listing 4: Model training and hyperparameter tuning with Keras and KerasTuner

```
1   # https://keras.io/api/keras_tuner/tuners/hyperband/
2   tuner = kt.Hyperband(
3     hypermodel=build_model,
4     objective='val_loss',
5     max_epochs=100,
6     factor=2
7   )
8
```

```
  early_stop = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    mode='min',
    restore_best_weights=True,
    patience=10
  )

  tuner.search(
    X_train,
    X_train,
    epochs=100,
    batch_size=300,
    verbose=1,
    callbacks=[early_stop],
    validation_data=(X_val, X_val)
  )
```

Listing 5: Hyperband search and early stopping setup

```
  best_hp = tuner.get_best_hyperparameters()[0]
  model = tuner.hypermodel.build(best_hp)

  history = model.fit(
    X_train,
    X_train,
    epochs=100,
    batch_size=300,
    verbose=1,
    callbacks=[early_stop],
    validation_data=(X_val, X_val)
  )
```

Listing 6: Best model re-training

```
  import matplotlib.pyplot as plt
  import seaborn as sns

  def get_x(metric):
    return range(len(history.history[metric]))

  def get_y(metric):
    return history.history[metric]

  sns.lineplot(
    history.history,
    x=get_x('loss'),
    y=get_y('loss'),
    label='Training'
  );
```

```
  sns.lineplot(
    history.history,
    x=get_x('val_loss'),
    y=get_y('val_loss'),
    label='Validation'
  );
  plt.xlabel('Epochs')
  plt.ylabel('Loss')
  plt.title('Binary cross-entropy loss')
  plt.ylim(0, 0.0020)
```

Listing 7: Binary cross-entropy loss visualisation

### 9.1.3 Embeddings

```
  embeddings = autoencoder.get_layer('code')(data)
  embeddings_df = pd.DataFrame(embeddings, index=data.index)
  embeddings_df.to_parquet(f'{model_name}.parquet', engine='
   pyarrow', compression='brotli')
```

Listing 8: Embeddings generation

### 9.1.4 Recommendations

```
  import pandas as pd
  from sklearn.metrics.pairwise import cosine_similarity

  # Loads the catalogue
  catalogue = pd.read_parquet('catalogue.parquet')
  # Gets the programmes PIDs (Top-Level Editorial Objects)
  tleo = catalogue.tleo_pid.values
  # Loads the embeddings
  embeddings = pd.read_parquet(f'{model_name}.parquet')
  # Gets the TLEOs embeddings
  tleo_embeddings = embeddings[embeddings.index.isin(tleo)]
  # Generates the similarity matrix
  similarity = pd.DataFrame(cosine_similarity(tleo_embeddings
   ), index=tleo_embeddings.index, columns=tleo_embeddings.
   index)
```

Listing 9: Cosine similarity calculation

## 9.2 Figures and tables

### 9.2.1 Passport tags

```
8 Passport predicates with the following unique value numbers:

- genre: 216
- format: 50
- contributor: 1654
- editorialTone: 54
- about: 6827
- relevantTo: 59
- motivation: 13
- narrativeTheme: 109

8982 features
```

Figure 13: Number of unique value annotations per Passport tag

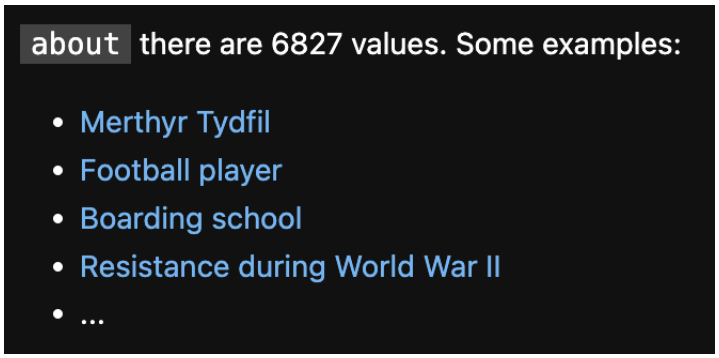



Figure 14: The "about" tag


Figure 15: The "contributor" tag

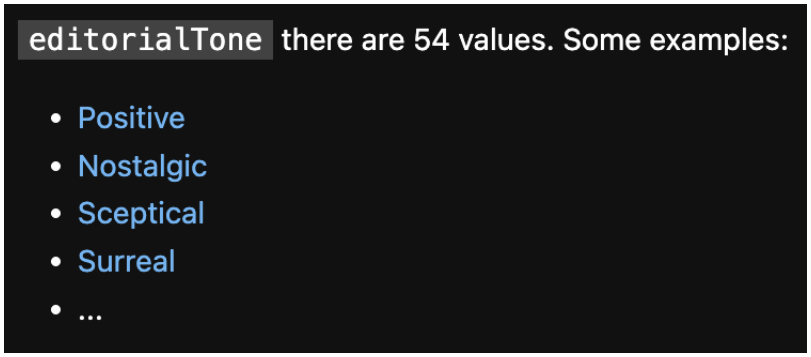



Figure 16: The "editorialTone" tag

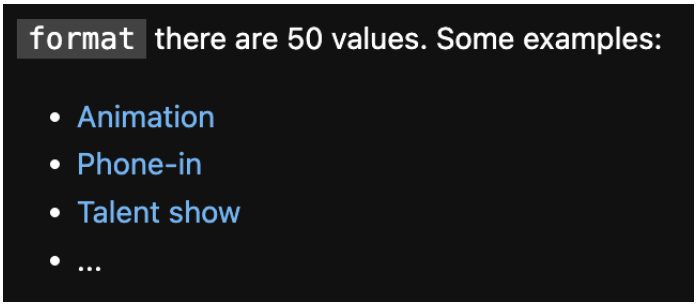



Figure 17: The "format" tag

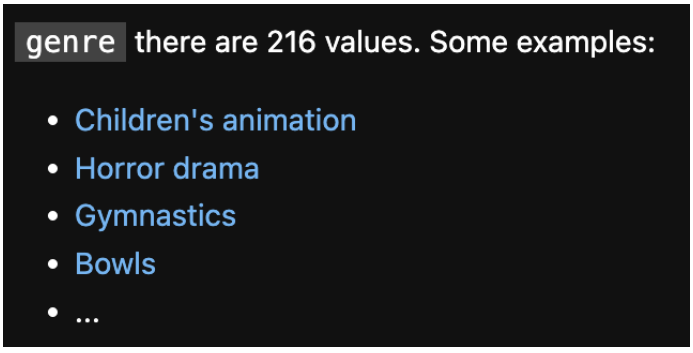



Figure 18: The "genre" tag

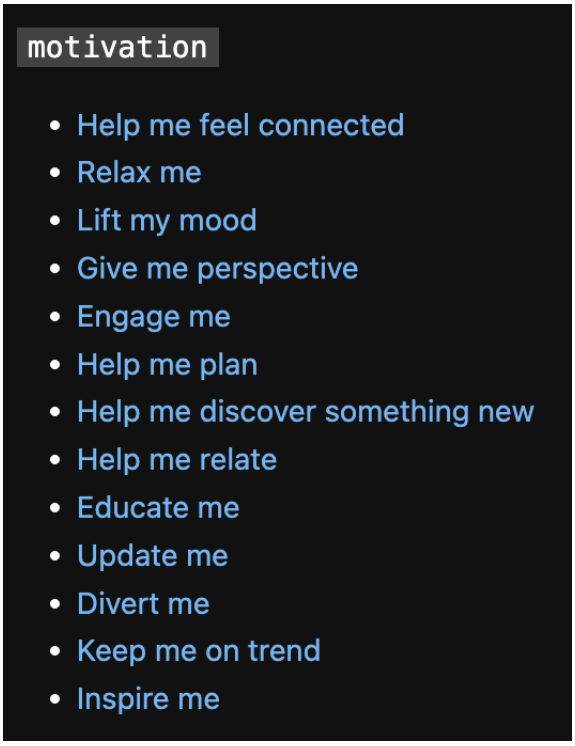



Figure 19: The "motivation" tag

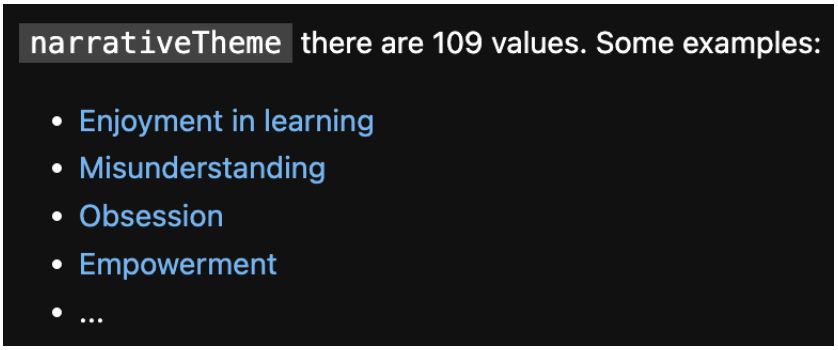



Figure 20: The "narrativeTheme" tag

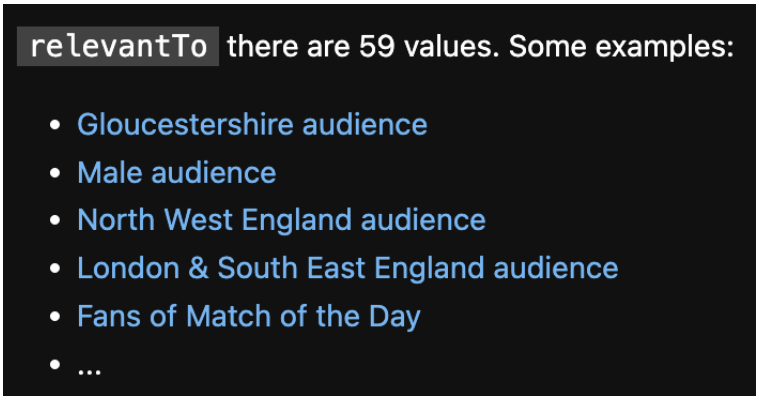



Figure 21: The "relevantTo" tag

### 9.2.2 Catalogue analysis

The following line plots visualise the daily time-series statistics of the BBC iPlayer catalogue used for training. New programmes are added to the catalogue, and unavailable ones are removed hourly. These updates change the size and tag composition of the catalogue and need to be monitored for data drift detection, retraining the model when big spikes occur.

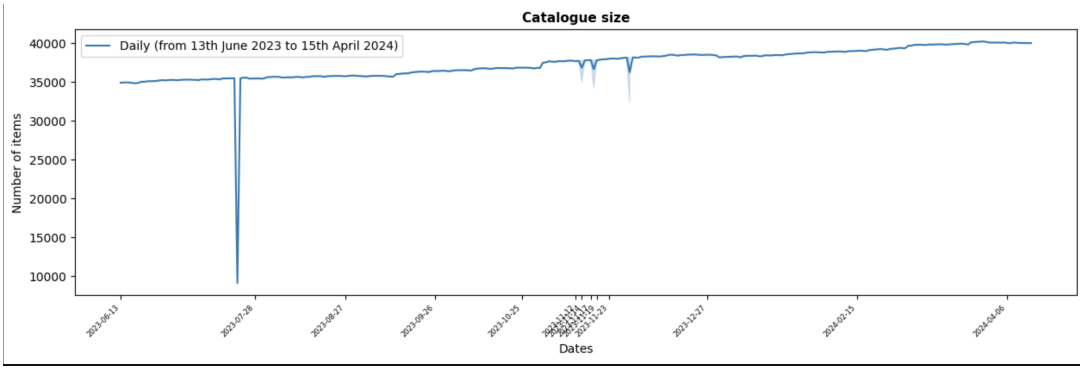



Figure 22: Daily catalogue size

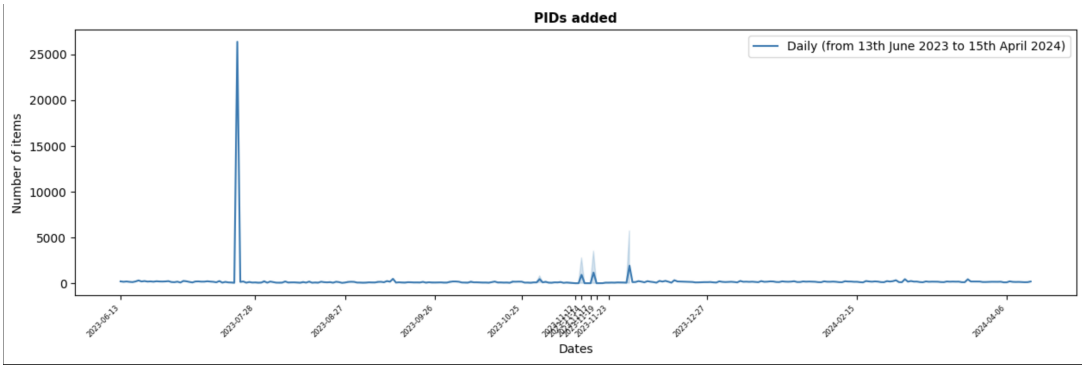



Figure 23: Number of PIDs added daily

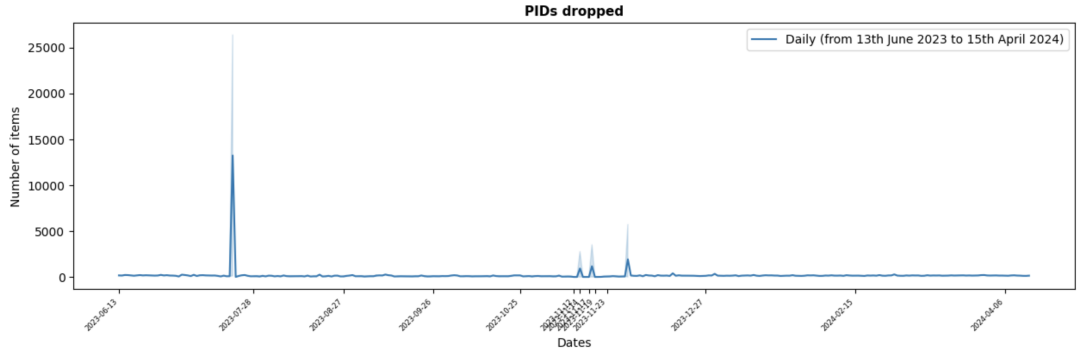



Figure 24: Number of PIDs dropped daily

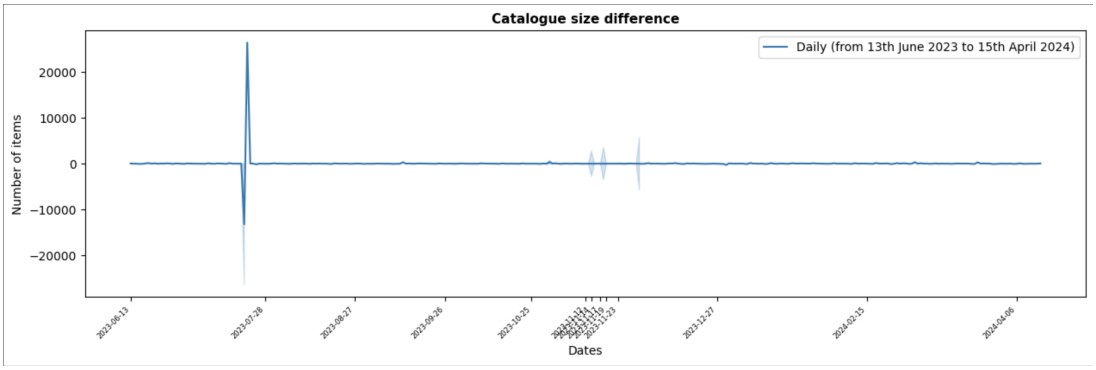



Figure 25: Daily catalogue size difference

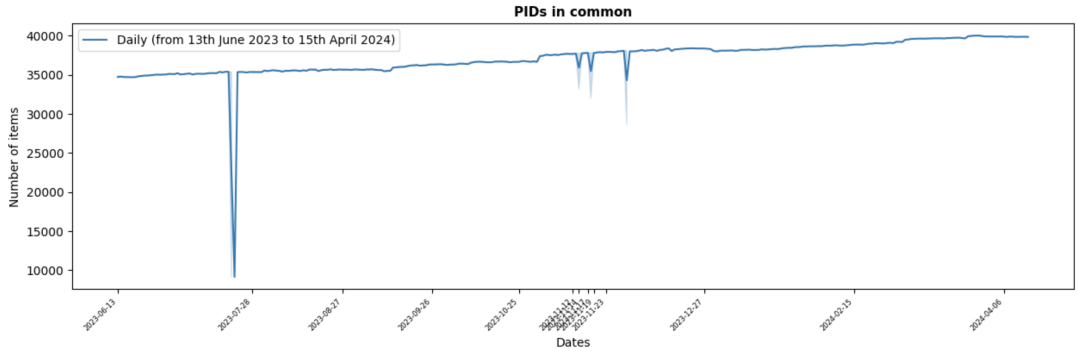



Figure 26: Number of PIDs in common daily

### 9.2.3 Programme structure and data distribution

The 18.19% of programmes have a *brand-episode* structure. An example is the "Christmas Celebration" show, presented by Sally Magnusson.

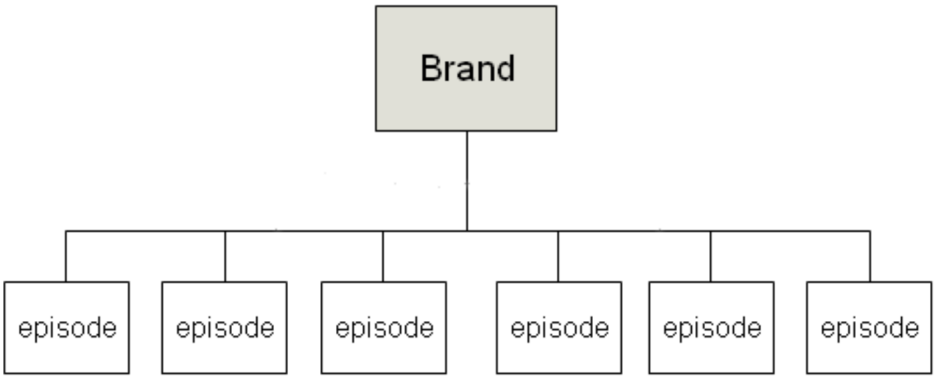



Figure 27: PIP Brand-Episode hierarchy

The 3.25% of programmes have a Series-Episode structure. An example is the four-part series "Inside Obama's White House".

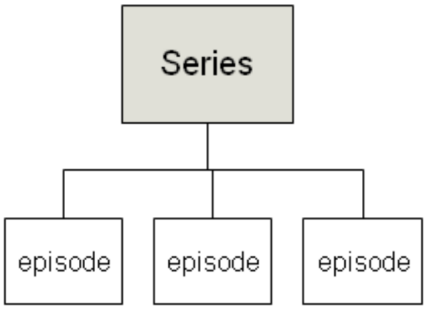



Figure 28: PIP Series-Episode hierarchy

The 75.12% of programmes have a Brand-Series-Episode structure. An example is the animated adventure comedy series "Go Jetters".

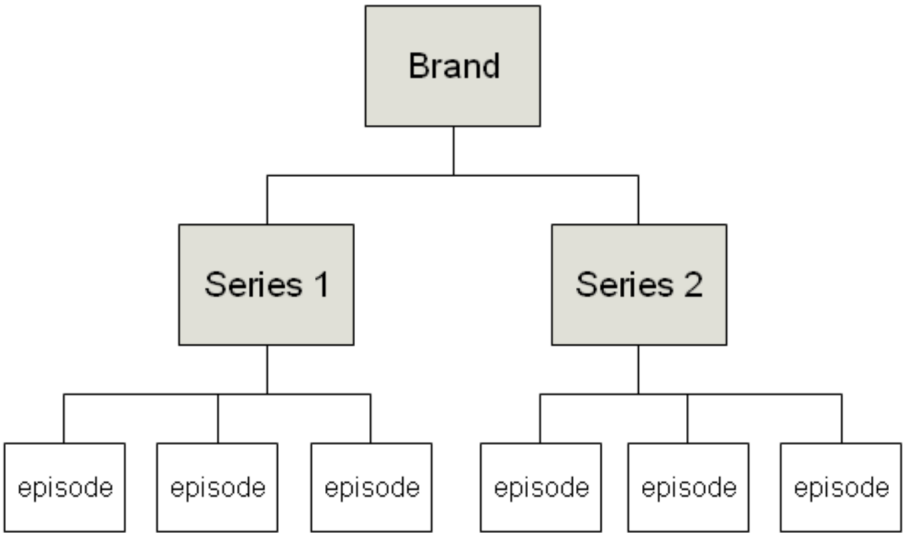
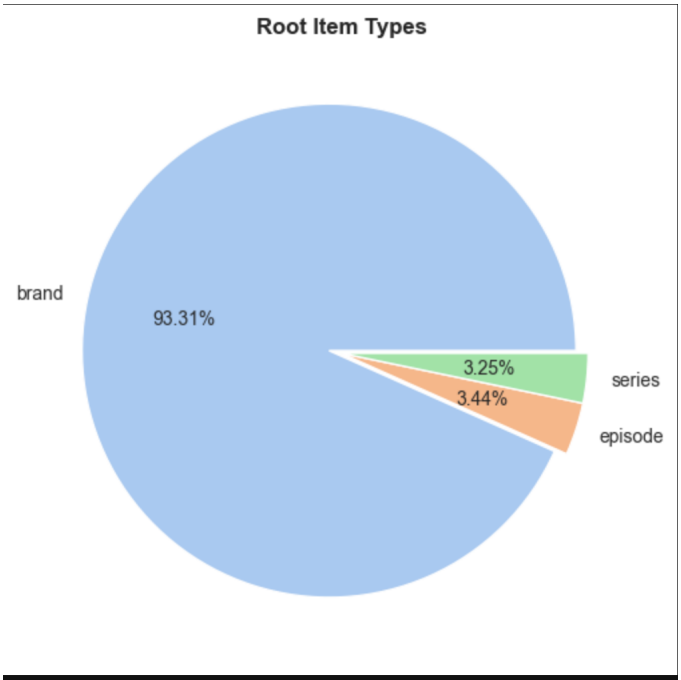


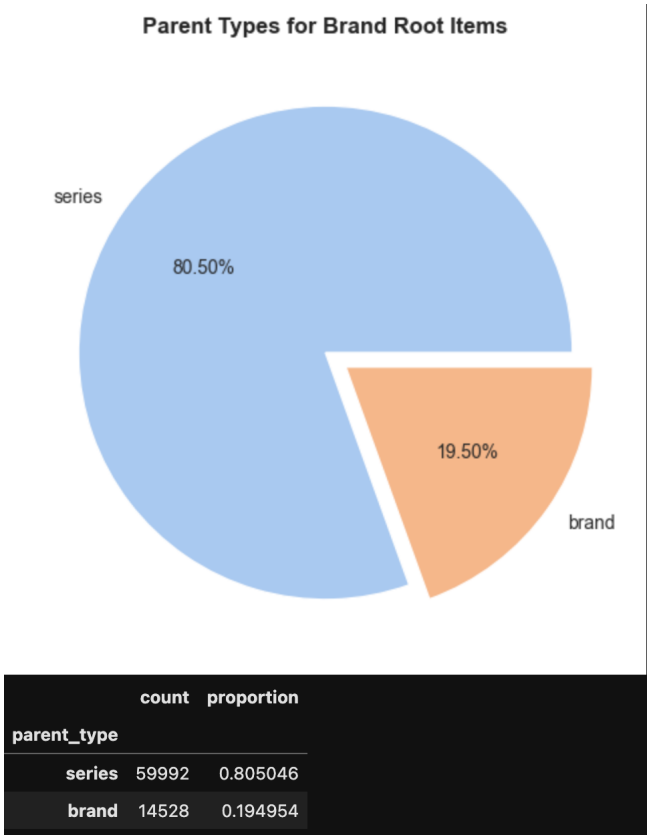

Figure 29: PIP Brand-Series-Episode hierarchy

The remaining 3.44% of programmes are *episode* only. An example is the comedy documentary "The Kemps: All Gold" about the Spandau Ballet brother's lives.




| tleo_type | count | proportion |
|---|---|---|
| brand | 74520 | 0.933086 |
| episode | 2745 | 0.034371 |
| series | 2599 | 0.032543 |

Figure 30: Root item types distribution




| parent_type | count | proportion |
|---|---|---|
| series | 59992 | 0.805046 |
| brand | 14528 | 0.194954 |

Figure 31: Parent types when the root item is a Brand

### 9.2.4 Recommendations

This is an example of quality testing using the visualisation tool, comparing the generated recommendations with the ones live on BBC iPlayer.

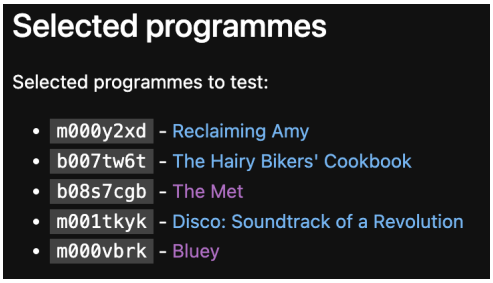



Figure 32: List of example programmes used for testing

Figure 33 shows the programme page for "Bluey" and the "More Like This" section on iPlayer, where 12 similar programmes are recommended.

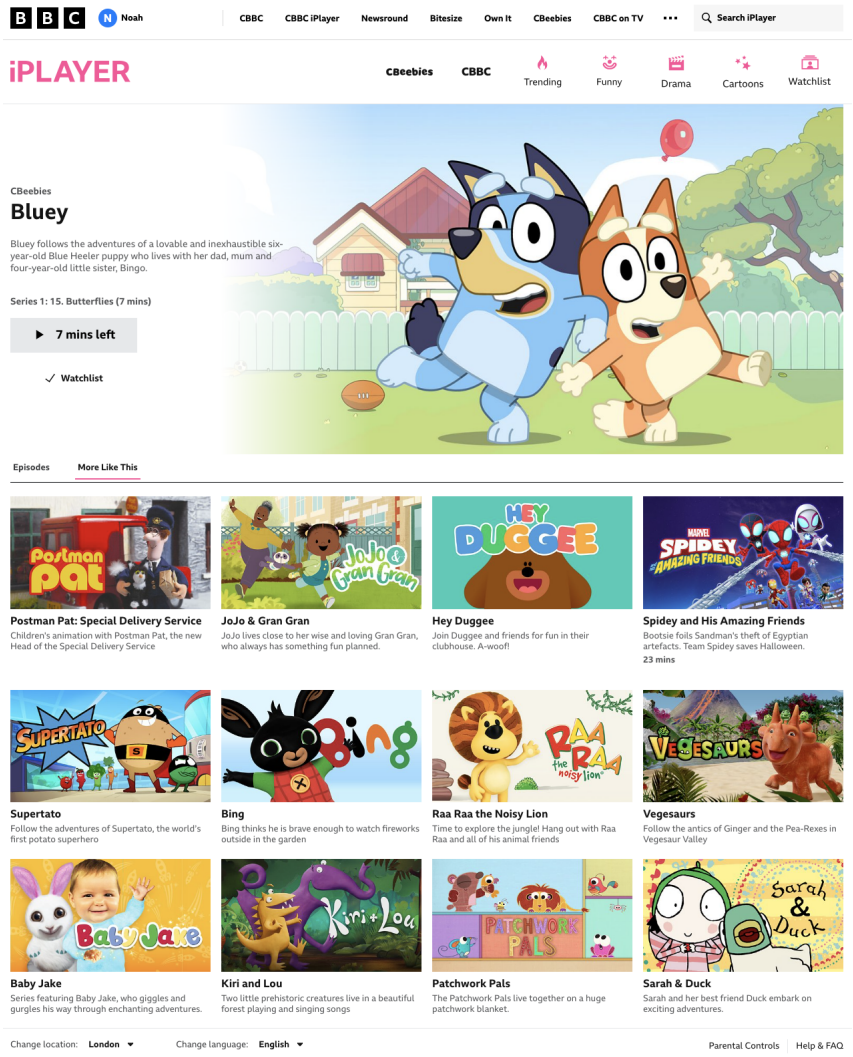



Figure 33: The "More Like This" section on BBC iPlayer

Figure 34 shows the seed programme for the top-12 content similarity recommendations.

Figure 34: “Bluey” (the seed programme)

The following figures show the recommendations generated by the pipeline in descending order of similarity score (the percentage in the parenthesis).

Figure 35: “Patchwork Pals” (96.017%)

Figure 36: “Timmy Time” (94.737%)

Figure 37: “Fireman Sam” (94.199%)

Figure 38: “Arthur” (94.061%)


Figure 39: “Postman Pat: Special Delivery Service” (93.940%)

Figure 40: “Tish Tash” (93.734%)

Figure 41: “Octonauts” (93.241%)

Figure 42: “Hey Duggee” (93.237%)

Figure 43: “Bob the Builder” (93.012%)

Figure 44: “Tee and Mo” (92.761%)

Figure 45: “Raa Raa the Noisy Lion” (92.487%)

Figure 46: “Mr Bear’s Christmas” (92.303%)

## 9.3 Mapping of the project report to the pass criteria

### Project Checklist Table

This is to cross check the project work completed by the apprentice meets the KSBs required by this assessment method.

### Pass Criteria

| **Awareness of the opportunities of AI and data science to create business value and growth.** (K13,K14) | | |
|---|---|---|
| | | **Project Mapping** |
| AI and data science solution developed within the project addresses a business need in line with quality standards and timescales. The business value of a data product / solution and any constraints making trade-offs accordingly have been considered. | | Section 2 (Outline of the issue, opportunity and the business problem to be solved) |
| **Critically evaluate the effectiveness and performance of proposed AI and data science solutions** (K23, S3, S17) | | |
| | | **Project Mapping** |
| Critically evaluates the performance of developed AI and machine models and the steps taken to mitigate sources of error and bias. | | Section 3.2 (modelling and regularisation in Methods and justification)<br><br>Section 5: Data selection (page 10 and 11)<br><br>Section 7 |
| Considers and selects from a range of appropriate principles, techniques and solutions to enhance the robustness of decisions at all stages. | | Section 3 (Methods and justification)<br><br>Section 6 (Survey of potential alternatives) |
| Critically evaluates the arguments, assumptions, abstract concepts and data to make | | Section 8 (Results, Summary of findings and recommendations, Implications)<br><br>, |

Figure 47: Project mapping criteria (page 1)

| | | |
|---|---|---|
| business focused recommendations. | | |
| Demonstrates how, from the range of possible solutions presented, they contributed to identifying the optimal solution. | | Section 6 (Survey of potential alternatives) |
| Explains how they implement data curation and data quality controls in line with organisational and regulatory requirements. | | Section 5 (Data selection, collection and pre-processing) |
| **Apply systematic methodology and project management principles in the delivery of innovative, stable and robust solutions (S2, S9, S10, S22, S25)** | | |
| | | **Project Mapping** |
| Selects and uses datasets, programming languages, tools and scientific methodologies to research business problems, providing a clear justification for their selection. | | Section 3 (Methods and justification)<br><br>Section 5 (Data selection, collection and pre-processing) |
| Analyses and critically evaluates test data and proposed solutions, considering current and future business requirements. | | Section 2 (Outline of the issue, opportunity and the business problem to be solved)<br><br>Section 4 (Scope of the project and Key Performance Indicators)<br><br>Section 7 (Implementation and performance metrics)<br><br>Section 8.1 (Results) |
| Manipulates and analyses complex datasets and critically evaluates arguments, assumptions, abstract concepts and data (that may be incomplete) to make recommendations and to enable a business solution or range of solutions to be achieved. | | Section 5 (Data selection, collection and pre-processing) |

Figure 48: Project mapping criteria (page 2)

| **AI Project and Development Management** (K6, S24) | |
|---|---|
| | **Project Mapping** |
| Correctly selects and applies development, research methodology and project management techniques to engage with customers and solve the business problem being addressed. | Section 1 and 2<br><br>Also, I did the following:<br><br>The project management style used was Agile. The team I worked in attachment with borrowed a bit from Kanban and a bit from Scrum.<br><br>To stay in line with the stakeholders needs, my work was always visible on the Kanban board in Jira. Every discovery, communication, etc. was always commented on the ticket for visibility and knowledge share. I was also in direct contact with stakeholders on direct messaging and emails. I also demoed the project to them at different stages.<br><br>I structured my research, by first investigating the data source. I spent some time understanding the data schema, where to pull the data from, the procedure for doing so, the freshness of the data (batch update vs near-realtime async updates) and who the people in charge of the data were, so to have a direct relationship and better insight. I then spent some time investigating the iPlayer catalogue. This gave me a big insight in how the data is structured, and learned a lot of quirks in the data you must be aware of, so your pre-processing can account for them. I dedicated an entire investigation on a Jupyter notebook where I pulled all my descriptive statistical knowledge and visualisation skills to highlight the findings.<br><br>Once I had a clear picture of the data, I started investigating the modelling, hyperparameter tuning and the generation of the first C2C recommender to visualise and compare the results with the existing model. |
| **Use of communication and influencing skills across teams** (K28, S4, S5, S7, S27, B2, B6) | |
| | **Project Mapping** |
| Describes how they have worked with a range of technical and non-technical stakeholders adapting their approach successfully to meet their diverse needs. | Section 4 (Scope of the project and KPI)<br><br>Section 8.1 (results) |
| Explains how to work autonomously and collaboratively with multidisciplinary teams indicating when each would be appropriate. | Section 4 (Scope of the project and KPI)<br><br>Section 8.1 (results) |

Figure 49: Project mapping criteria (page 3)

| | | |
|---|---|---|
| Describes how they have analysed information and data, using questioning and discussions with subject matter experts to scope new AI and data science requirements. | | Section 4 (Scope of the project and KPI)<br><br>Section 8.1 (results) |
| Written and verbal communication is clear, structured and appropriate for the audience. | | Section 4 (Scope of the project and KPI)<br><br>Section 8.1 (results) |
| Explains how to work with software engineers to ensure suitable testing and documentation processes are implemented. | | Section 3.4 (Tools and Frameworks)<br><br>Section 8.1 (results) |
| **Application of technical knowledge** (K1, K3, K5, K26, S11, S15, S18) | | |
| | | **Project Mapping** |
| Describes how they applied appropriate scientific and technological methods for machine learning, AI and data science solutions, services and platforms to deliver business outcomes outlining successes and challenges. | | Section 3 (Methods and Justifications)<br><br>Section 5 (Data selection, collection and pre-processing)<br><br>Section 6 (Survey of potential alternatives)<br><br>Section 7 (Implementation and performance metrics) |

## Distinction Criteria

| **Awareness of the opportunities of AI and data science to create business value and growth.** (K13,K14) | | |
|---|---|---|
| | | **Project Mapping** |
| Articulates a commercial awareness of organisational priorities. Explains how the | | Section 1 (Introduction and Background) |

Figure 50: Project mapping criteria (page 4)

| | | |
|---|---|---|
| practical trade-offs in implementing an AI or data science solution for the particular business context have been addressed and shape the solution accordingly to optimise outcomes. | | Section 2 (Outline of the issue, opportunity and business problem to be solved)<br><br>Section 5, page 8 (trade-off about not integrating with UCED) |

Figure 51: Project mapping criteria (page 5)

| **Critically evaluate the effectiveness and performance of proposed AI and data science solutions** (K23, S3, S17) | |
|---|---|
| | **Project Mapping** |
| Critically evaluates and adapts practice making recommendations for communicating technical methodology. | Section 8.1 (Results) |
| Explains when they have effectively communicated technical information in a team context which has influenced others and impacted positively on decisions or working practices. | Section 8.1 (Results)<br><br>Section 8.3 (Implications)<br><br>To honestly answer this I asked for an external feedback. In the words of one of the senior Data Scientists:<br><br>"I think you can honestly talk about making inroads into incorporating passports data into our working practices. The first iteration of that was helping the iRex team (or the new c2c squad) better understand:1) the value of using passports data due to its richness in terms of tags & also high coverage; 2) how items similarities trained on Autoencoder produce recommendable items of high quality as demoed to the CD team." |
| **AI Project and Development Management** (K6, S24) | |
| | **Project Mapping** |
| Can evidence suitable methodology and tools have been selected with understanding of the impact of this choice on working practice, along with the risks to continuity of working practice that may arise if such solutions are not utilised. | Section 3.4 (Tools and frameworks)<br><br>From a methodology point of view:<br><br>I decided to use an Agile methodology to manage this project because this is how I work daily but also because it is an effective way to manage a project with minimal overhead. It allows to react to changes quickly.<br><br>If I had used a more "waterfall" approach, I would have had to produce an extensive and detailed documentation filled with diagrams about something I didn't even investigate beforehand. Also, my understanding of the project, domain and hence of the possible solutions changed many times overtime. This meant that if I hadn't used Agile, I would have wasted a lot of time before realising that I needed to adapt to something I didn't plan. |

Figure 52: Project mapping criteria (page 6)

| **Use of communication and influencing skills across teams**<br>(K28, S4, S5, S7, S27, B2, B6) | |
|---|---|
| | **Project Mapping** |
| Explains how they adapted their approach with a range of technical and non-technical stakeholders and in different situations in order to achieve the best outcome for the business. | Section 8.1 (Results)<br><br>The main differentiator in approach when I dealt with technical and non-technical stakeholders was the focus of the conversation.<br><br>With technical stakeholders I was focused on conveying the approach on a technical basis and I made sure to communicate the different options considered for a specific solution, to give the opportunity to assess them themselves and to ask further questions. This technique had two advantages: it built more trust and confidence with the adopted solution because it can be peer reviewed. It was also an opportunity to improve upon it if a better idea was floated during the conversation.<br><br>With non-technical stakeholders I was focused on the outcome of the solution. Editorials, product owners or executive have other priorities, and it was imperative the solution I wanted to build was properly communicated within the context of what it was meant to solve and what Key Performance Indicators would positively impact. |
| Evaluates solutions and explains the risks and implications of the AI data science requirements and alternative approaches and ways to address them. | Section 8.1 and 8.4 |
| **Application of technical knowledge**<br>(K1, K3, K5, K26, S11, S15, S18) | |
| | **Project Mapping** |
| Explains the rationale for selecting particular technical solutions, including the relevant consideration of scientific benefit and suitability for working practices. | Section 3, 5, 6 and 7 |
| Appraises AI and/or Data solutions and explains the risks and implications of the process, alternative approaches and ways to address them. | Section 6<br><br>Section 8.4 |

Figure 53: Project mapping criteria (page 7)

# References


[1] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *CoRR*, abs/2003.05991, 2020.

[2] BBC. BBC Ontologies. https://www.bbc.co.uk/ontologies/.

[3] BBC. Bluey - "more like this" tab. https://www.bbc.co.uk/iplayer/episodes/m000vbrk/bluey?seriesId=more-like-this.

[4] BBC. Programme Pages. https://downloads.bbc.co.uk/academy/collegeoftechnology/docs/j000m977x/iB2_programme_pages.pdf.

[5] BBC. Things. https://www.bbc.co.uk/things.

[6] BBC. Things - About. https://www.bbc.co.uk/things/about.

[7] BBC. Things - API. https://www.bbc.co.uk/things/api.

[8] BBC. The Royal Charter. https://www.bbc.com/aboutthebbc/governance/charter, 2017. Valid until 31 December 2027.

[9] BBC. How metadata will drive content discovery for the bbc online. https://www.bbc.co.uk/webarchive/https%3A%2F%2Fwww.bbc.co.uk%2Fblogs%2Finternet%2Fentries%2Feacbb071-d471-4d85-ba9d-938c0c800d0b, 2020. This page was archived on 1st August 2023 and is no longer updated.

[10] Tomislav Duricic, Dominik Kowald, Emanuel Lacic, and Elisabeth Lex. Beyond-accuracy: A review on diversity, serendipity and fairness in recommender systems based on graph neural networks, 2023.

[11] Google. Embeddings. https://developers.google.com/machine-learning/crash-course/embeddings.

[12] UK Government. UK GDPR. https://www.legislation.gov.uk/eur/2016/679/contents, 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council.

[13] Marius Kaminskas and Derek G. Bridge. Diversity, serendipity, novelty, and coverage. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 7:1 – 42, 2016.

[14] Lisha Li, Kevin G. Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *CoRR*, abs/1603.06560, 2016.

[15] Umberto Michelucci. An introduction to autoencoders. *CoRR*, abs/2201.03898, 2022.

[16] pandas via NumFOCUS, Inc. MultiIndex / advanced indexing. `https://pandas.pydata.org/docs/user_guide/advanced.html`, 2024.

[17] Simone Spaccarotella. Recommendation assumptions. `https://www.slideshare.net/slideshow/recommendations-assumptions/236291920`, 2015. Presented at Prototyping Day @ Mozilla London on 3 September 2015, at Engineering Summit @ BBC on 7 March 2018, uploaded on SlideShare on 27 June 2020.

[18] Michael Smethurst. Designing a URL structure for BBC programmes. `https://smethur.st/posts/176135860`, 2014. Smethurst blog post published on Sep 25, 2014.

[19] W3C. RDF 1.1 Concepts and Abstract Syntax. `https://www.w3.org/TR/rdf11-concepts`, 2014. W3C Recommendation 25 February 2014.

[20] W3C. RDF 1.1 Turtle - Terse RDF Triple Language. `https://www.w3.org/TR/turtle/`, 2014. W3C Recommendation 25 February 2014.

[21] W3C. Resource Description Framework (RDF). `https://www.w3.org/RDF`, 2014. Publication date: 2014-02-25 (with a previous version published at: 2004-02-10).