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

Simone Spaccarotella

September 2024

Contents

1 Introduction and background

I am a Software Engineer at the BBC, Team Lead for the Sounds web team, and I have been training as a Data Scientist, working in attachment with the iPlayer Recommendation team.

I built a machine learning model pipeline that produces content-to-content (C2C) similarity recommendations of video on-demand (VOD), for the "More Like This" section on BBC iPlayer [2]. 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, and it sparked an interest. I had a tangent encounter back in 2015 while working for a team that was building an initial recommender for BBC News, and an API to provide recommendations using 3rd party engines. During a Hack Day some time later, I produced and presented a talk called "Recommendation Assumptions" [12], which was about types of recommendations and external factors affecting them, which are contextual to the consumption of the content itself. Until these days, where I was able to finally put my knowledge into practice with an actual project on real data.

The BBC is a well-known British broadcaster, and it is always 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 [6]. The current media landscape requires the BBC to deliver digital-first content that is relevant to the audience, and this involves investements in data and personalised services, not to mention a certain revolution in machine learning that is keeping everyone busy.

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

The BBC produces and stores a vast amount of data for its content, and this data is produced and surfaced by countless services and APIs. One of the priority for the BBC is to increae the usage across the business of "Passport" tags [7] an internal BBC system that provides a richer set of metadata annotations for multi modal content (audio, video and text). The usage in production is very low, and its BBC-wide adoption would make the access to metadata consistent, removing duplications and reducing effort and costs.

Furthermore, the similarity score of the current C2C recommender is directly proportional to the number of values in common between any pairs of items on a per-feature basis. But the commonality is calculated with an exact string equality, hence it ignores any relationship between different categorical values expressing a similar concept (e.g. "comedy", "stand-up comedy").

Moreover, the number and types of tags are not enough to sufficiently describe the content, while the data distribution is severely skewed towards the most popular annotations and no pre-processing is applied.

Lastly, each similarity score is multiplied by a hardcoded weight that modulates the importance of a feature, but it doesn't solve the polarising effect of a skewed distribution. Unfortunately, because they are hyperparameters and not learned weights, the model can't improve its performances by minimising them against a cost function.

To address these issues, the aim of this project was:

- **To improve the quality of the C2C similarity recommendations**. The hypothesis was that by using in input a richer set of metadata that better describes the content, and by reducing the high-dimensional data to a lower-dimensional latent manifold, the model would be able to generate embeddings that could improve the quality of the recommendations, by mapping the item similarity problem to a geometric distance calculation between vectors in a multi-dimensional Euclidean space.

- **To reduce the costs to generate C2C similarity recommendations**. I used Passport tags as input to build a general solution that could be used by any product and applied to any type of content, because they all share the same set of common annotations.

- **To build a foundational item-embeddings generator**. Content-based recommenders use item metadata. This project provided an immediate solution for C2C unpersonalised recommendations that solely relies on them and it could provide a foundational approach for personalised recommender that combine content metadata with other signals like user interactions and contextual data.

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. It's a simple yet effective encoding method and it's perfect for the transformation of nominal categoricals, because it doesn't introduce any ranking and/or arithmetic relationship among the encoded values. The downside of this approach is that it generates high-dimensional sparse arrays, introducing the so called *curse of dimensionality* problem. Nonetheless, this was an accepted drawback which was managed in the modelling phase.

3.2 Modeling and regularisation

I trained an undercomplete **autoencoder** to improve the computational complexity and the quality of the output at inference time. Autoencoders are self-supervised models, capable of capturing non-linearity from the data. This type of encoder-decoder limits the number of nodes in the hidden layers, and creates a "bottleneck" of information flow through the neural network. This bottleneck structure is a form of *regularisation* that forces the model to learn latent attributes from the input, while reconstructing it with minimal loss. Ultimately, it prevents the model from *overfitting* the training dataset, by indexing it like a caching layer.

The *encoder* part of the trained encoder-decoder is used to compress the one-hot encoded high-dimensional sparse vectors into so-called *embeddings*, a lower-dimensional dense representation of the initial content metadata. This technique solved the curse of dimensionality and the data sparsity problems, and improved the calculation complexity and quality during inference.

To improve and assess the ability of the model to *generalise* on unseen data, I randomly shuffled the dataset and split it into 3 chunks: training, validation and test. I run **hyperparameters tuning** to find the best set of model parameters that minimised the objective function and I 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 certain "patience" threshold, after reaching a local minima on the validation error.

I used **dropout** to further regularise the model and make it robust to small changes in the input, and **data augmentation**, by including in the training data the episodes of a programme that share the same tags with their parent container.

Weight decay and **batch normalisation** were tested during hyperparameter tuning and discarded for poor performances.

3.3 Inference

Item similarity was calculated with the **cosine** of the angle θ between each pair of embeddings. This metric is insensitive to the magnitude of the vectors, and because popular values tend to have a larger magnitude, it mitigates the impact of popularity in the similarity calculation. In addition, multi-hot encoded pairs can only have a finite number of angles. They represent unit vectors bound in the "positive quadrant" of a multi-dimensional Euclidean space. This forces the cosine similarity to also assume a finite number of discrete values between 0 and 1, leading to an information loss. Ideally, we would expect the similarity score to assume a continuous value bound between -1 and 1, and this is only possible if the angle θ of any vector pair can assume a continuous value between 0 and 360 (i.e. 0π and 2π), 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 facilitated 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 it wasn't possible to use it for exploratory data analysis (EDA), because the iPlayer catalogue used in development had roughly one-year worth of data and it didn't fit in memory, causing Pandas to crash. For this reason I used **Dask**. It is a library capable of running out-of-memory and parallel execution for faster processing on single-node machines and distributed computing on multi-nodes 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 tuning. 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.

For data visualisation I used a combination of **Matplotlib** and **Seaborn**, while I used **rdflib** to fetch and parse the RDF documents from the BBC Ontology. These documents represented the metadata values and contained

the actual entity label. This label was needed to hydrate the set of metadata for each item, to visualise the recommendations for testing purposes. Worth also 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

The scope for this project was to build an end-to-end machine learning solution able to produce unpersonalised content-to-content similarity recommendations, using Passport metadata tags as input. To measure success, I defined a set of key performance indicators (KPIs).

The minimum viable outcome for this solution was to produce a set of recommendations comparable with the ones currently in production and as a desired outcome to increase user engagement. The ability to do so using Passport tags and the ability to be a general solution applicable to multimodal BBC content were also accounted as KPIs.

Comparability was a qualitative and subjective KPI that served as a compass, indicating whether the project was progressing towards the right direction. It was evaluated by several technical and non-technical stakeholders, with a diverse domain knowledge and background. I built a rudimentary visualisation tool that rendered the title, image, description and metadata of the seed programme and the top-k C2C recommendations. People involved could give their subjective feedback on what was their perceived level of similarity of the output, testing edge cases and common use cases with expected outcome, sensitive recommendations (i.e. content recommendations for children), and discussing whether there were anomalies, expected, unexpected and/or surprising results.

To measure user engagement though, the system needed to be A/B tested in production with real data. Unfortunately, there were too many moving parts outside my control that needed to happen first, for me to be able to measure this KPI. It would have delayed the project, increasing the chance of failure. To mitigate this risk, I defined a hypothesis supported by a KPI that could be measured offline within my area of control, moving the testing of this hypothesis outside the scope of the project.

The hypothesis stated that long-term user engagement is impacted by the degree of diversity of recommended content. It went on sayng that a diverse set of recommendations can generate new and unexpected results, which can increase surprise and serendipity, pushing the user away from boredom. This was an untested hypothesis, but grounded in active research on the topic, such as [10] and [8], that supported the initial statement. Although the hypothesis needed validation, it was ok for it to be pushed out of scope, because it wasn't too far fetched afterall. For this reason, I defined a proxy metric for diversity that was used to measure an approximation of it offline, pending a future A/B test validation.

5 Data selection, collection and pre-processing

News and Sports articles, iPlayer videos on-demand, Sounds podcasts etc. are annotated with the so-called Passport tags. These tags describe the content produced by the BBC and can be used for retrieval (search) and filtering (recommendations). They can be applied either 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, a self-service metadata delivery platform that exposes data as a document stream for products to integrate with. This platform 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. These objects in turn contain two properties: "predicate" and "value". They represent the name and the value of a tag, and are expressed as URL-formatted strings, with the exception of dates. The predicate is a class of the BBC Ontology [1] while the value can be a date or an entity defined as an RDF [15, 13] document, accessible in Turtle format [14] via the BBC Things API [3, 4, 5]. These entities are linked to each other and/or to external resources, and are described by attributes and relationships, giving the data a graph structure.

For the development of the project, I decided not to integrate with UCED but to use batches of Passport files, manually collected and stored on a local folder. This was a tradeoff that allowed me to develop the project with real data while keeping the costs down, given that the resources needed to be set on two AWS accounts. Moreover, I didn't want to pass the burden of maintenance to the team that owned the accounts, without having tested the feasibility of the solution first.

Content metadata is not classified as personally identifiable information (PII), according to the UK GDPR [9]. Nonetheless, this data is regulated by internal BBC data governance policies and as such, it is encrypted at rest and in transit by default. For this reason, no further precautions were required during storage and processing.

I chose to use Passport because this data provides a set of tags shared across all content produced by the BBC, making this a general solution that reduces duplications and ultimately costs. Passport provides a flexible and rich set of tags to describe any content. Annotations can describe canonical information such as *genre* and *format*, but also things like the "contributor" that features in, the "narrative theme", the "editorial tone", what the content is "about" or what relevant *entities* are mentioned, etc.

During pre-processing, a list of JSON files were loaded into the pipeline and the tags extrapolated into a dictionary data structure. The key of the dictionary was the programme ID (called *PID*) and the value was another dictionary describing the annotations. A programme can be tagged with the same predicate multiple times, as long as it has different values, while the same value (e.g. "Music") can be used by multiple predicates (e.g. "about" or "genre").

The dictionary was transformed in a Pandas *Dataframe*, where the rows represented the programmes and the columns the tags. I used a MultiIndex [11] for the columns because I needed to keep the duplicate values (2nd-level index) across the predicates (1st-level index). I then populated the cells of the *Dataframe* with the value `"1"` if the programme was annotated with the corresponding tag, or `"0"` otherwise. This process generated one-hot encoded arrays. I initially started with a vectorisation approach known to be performant, using the `"get_dummies"` Pandas function. Surprisingly, it was slower and less scalable of the solution that I adopted in the end.

A source of bias in the dataset was the usage of the `"mentions"` tag. This tag 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. Something "mentioned" doesn't necessarily describe what the content is about, because of the intrinsic ambiguities of natural languages. Figure of speech devices such as metaphors, analogies, allegories, etc., alter the meaning of a sentence for stylistic effect and can affect the representation of what the content was about. For example, if the idiom "being over the moon" is mentioned referring to something unrelated with space, and the term "moon" is extracted as a descriptor, it certainly increases the chances of misrepresentation. To mitigate this source of bias I dropped the tag in favour of `"about"`, a tag that describes what the content is really about. This tag is manually annotated by editorial teams who are trained to only annotate with relevant and topical entities.

A source of error was the encoding of unseen tags. I decided to drop the new tags and encode the data with the existing ones only. Also, a subset of programmes didn't have any annotations. Creating an entry for them would have generated a minority of one-hot encoded arrays with all zeros, representing a characteristic class of uninformative observations. I decided to drop these programmes and include the only the ones with at least one annotation.

6 Survey of potential alternatives

7 Implementation and performance metrics

8 Results

9 Discussion & conclusions/recommendations

10 Summary of findings

11 Implications

12 Caveats & limitations

13 Appendices

References

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

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

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

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

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

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

[7] 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.

[8] 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.

[9] 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.

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

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

[12] 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.

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

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

[15] 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).