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 [1]. 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" [4], 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 [2]. 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 metadata for its content, and this metadata is surfaced by countless services and APIs. One of the priority for the BBC is to increae the adoption across the business of "Passport" [3], an internal service that generates, stores, manages and provides access to a richer dataset 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 sourced the input data from Passport, 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 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 & justification

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 into the modelling phase. Nonetheless, this was an accepted drawback which was managed by the modelling phase of the pipeline.

I trained a regularised **undercomplete autoencoder** to reconstruct the data in input. The autoencoder is capable of capturing non-linearity in the data, learning latent key aspects that explain the underlying structure. The dataset was randomly shuffled and then split in three chunks: training, validation and test. The neural network was trained on the training set, while the validation set was used for **hyperparameter tuning**, monitoring the reconstruction loss on this chunk of data within a pre-defined "patience" treshold of epochs as a condition for the **early stopping**. The model with the best set of hyperparameters, was validated against the test set. Finally, I extracted the *encoder* part from the trained autoencoder, to generate embeddings of unseen metadata during inference, compressing the one-hot encoded high-dimensional sparse array into a lower-dimensional dense space, which mitigated the curse of dimensionality and the data sparsity.

The item similarity was calculated with the **cosine** of the angle θ between each pair of vector 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 vector pairs can only have a finite number of angles (i.e. zero, right angle or something in between) and are bound in the "positive quadrant" of a multi-dimensional Euclidean space. This situation forces the cosine similarity to also assume a finite number of discrete values between 0 and 1, leading to an information loss for item similarity. 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.

The entire project was written in **Python**, because it is the *de facto* programming language for data science and machine learning tasks, and it is also the language of choice at the BBC, which fosters collaboration and code maintenance. It also has an established, diverse and well-documented ecosystem

6

of external libraries and frameworks that facilitate the job.

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, using the familiar Pandas API.

For modelling I used **TensorFlow** and **Keras**, to build 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 (and other relationships between entities). 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. 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

References

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

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

[3] 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.

[4] 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.