Complete the Outfit with Boosted Trees

The 2nd place solution to SIGIR's 2022 Data Challenge hosted by FARFETCH

David Albrecht
San Diego, CA, USA
DavidPabloAlbrecht@gmail.com

ABSTRACT

The main goal of this challenge was to develop a model that could generate outfits for each individual product. This is a particularly challenging task, as the patterns that make an outfit acceptable are complex. Most of the properties that make products suitable to belong to the same outfit typically do not exist as product metadata: can a floral short dress be paired with white sandals and a yellow tote bag? The answer is: *I need to see the products first.*

To approximate this task contestants were asked to predict the item that was missing from a given outfit, also known as Fill in the Blank (FITB). The solution outlined here uses an ensemble of boosted trees in Microsoft's LightGBM library and achieves a FITB of 0.764 on the final test set.

KEYWORDS

Boosted Trees, Fashion, E-Commerce, FITB

1 Methodology

The solution's general methodology is as follows:

- Concatenate Fashion Outfits, Stage 1 test set, and Stage 2 test set.
- Generate N positive labels per outfit by removing each item.
- Generate 20 random negative labels per positive label.
- Process Product Metadata and feature engineering.
- Train 20 separate LightGBM classifiers on one of 20 disjoint data partitions, totaling the full training set.

Regarding data processing and feature engineering, contestants were provided with the following data sources:

- Product metadata: A table composed by approximately 400,000 products with product attributes, such as family, category, brand.
- Product images: Each product will have a single image with the item photographed with a frontal view in a white background. The images do not contain any people.
- Fashion outfits: Table with approximately 300,000 outfits. This was the main training set.

Product metadata was used for feature engineering. The below features were created in the following ways:

- full_category: Concatenate product_family, product_category, and product_sub_category. One hot encode.
- product_family: No cleaning, one hot encode.
- product_category: No cleaning, one hot encode.
- product_subcategory: No cleaning, one hot encode.
- product_gender: Lowercase, one hot encode.
- product_main_colour: Lowercase, one hot encode.
- product_second_color: Fill nulls, lowercase, one hot encode.
- product_brand: Lowercase, one hot encode.
- product_attributes: Fill nulls, lowercase, one hot encode.
- product_materials: Fill nulls, lowercase, one hot encode.
- product_image_path: Split on forward slash, one hot encode.
- product_highlights: Fill nulls, remove rare characters, keep tokens that occur 50 or more times, one hot encode.
- product_short_description: Use custom text cleaner to keep tokens that occur 50 or more times and are at least 2 characters in length, extract bigrams, one hot encode.
- popular_product_id: One hot encode IDs that occur 149 or more times (99.5th percentile of occurrences in all outfits).
- image_hash: 8x8 perceptual image hash.

After processing features per product, the one hot encodings per incomplete outfit, treated as a query, were summed and concatenated to the one hot encoding representation of the candidate product, treated as a proposed answer. This flat array was linked to the binary dependent variable, depending on whether the candidate was the generated positive or negative sample, and modeled using LightGBM's default parameters with exception of num_leaves (1,000) and n_estimators (1,000). Predictions were made on the test sets by summing scores equally from all models.

2 Results

The final FITB achieved was 0.764. Multiple iterations, each with more features and preprocessing, and different ensemble methods were tested to arrive at the current solution.

3 Future Work

Much of the performance of the final ensemble can be attributed to processing the short description feature. However, this simple approach can likely be improved by leveraging word embeddings

and either replacing the bag of words model or complimenting it. This may include a Word2Vec model or a large language model such as BERT, CLIP, or Fashion CLIP.

Similarly, pre-trained models such as ResNet/ImageNet or the CLIP models could prove useful to process images. These features were not included in the final model but were fine-tuned as a standalone model and did not perform as well as the LightGBM approach. Combining these models, however, in a two-stage fashion may improve results: By fine-tuning ResNet or CLIP models, one can extract N predictions and use them as features in the current LightGBM approach as a second stage in the classification pipeline.

Other, more minor, improvements include the following:

- Convert the one-hot bag of words model to a count of words or a TF-IDF representation.
- More random, negative sampling.
- More complex negative sampling with typical candidate generation such as collaborative filtering based on categories or other product metadata.
- Training a single model on all the data, as opposed to an ensemble which was necessary due to compute constraints.

Lastly, with such promising FITB scores, one considers that the overall challenge design could have been altered to align with the ultimate business need more closely, which is the ability to truly complete outfits. The current design does not fully lend itself to this objective for one reason: The candidates were generated for contestants. While this made the competition simpler, it also inflates the out-of-sample efficacy of any proposed solution in production with real users where the pool of candidates would be the entire product catalog. This design change allowed contestants, and this solution, to leverage both Stage 1 and Stage 2 test sets as training data and effectively introduce leakage that unfairly improved (albeit not significantly) the final FITB score. A proposed change would be to remove these candidates and instead force contestants to generate them. Further, as opposed to FITB, a metric that measures positions in a recommended list, like MAP@K, could more closely represent the experience users will encounter when interacting with the model.

4 Conclusion

LightGBM, and boosted trees in general, have proven to be powerful algorithms to model problems that are typically solved using neural networks but would otherwise be difficult given large, sparse feature spaces. Given more time, however, combining these approaches could have resulted in better FITB scores.

REFERENCES

[1] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". Advances in Neural Information Processing Systems 30 (NIPS 2017), pp. 3149-3157.

[2] Qi Meng, Guolin Ke, Taifeng Wang, Wei Chen, Qiwei Ye, Zhi-Ming Ma, Tie-Yan Liu. "A Communication-Efficient Parallel Algorithm for Decision Tree". Advances in Neural Information Processing Systems 29 (NIPS 2016), pp. 1279-1287.