Predicting Click-through Rates of Text Search Ads Using Handcrafted Features

Melvin Charles O. DY
OPT Inc. AI Solutions Development Dept. (株式会社オプトAIソリューション開発部)
m.dy@opt.ne.jp

Abstract

In this paper, we present the results of our work on predicting CTR from search ad texts using an implementation of handcrafted features, and their use in several regression models. Our approach is shown to have some advantages over an approach that uses only BERT-like learned features.

1 Introduction

1.1 Motivations

The question of optimizing for clicks is an evergreen matter of interest in the world of online advertising. While there are many factors involved in between the creation of ad materials and their final presentation to the end-user, including bidding for prime positions, and optimization on the platform side, the use of click-through rate (“CTR”) as a metric and objective variable for the entire process is generally accepted.

Many business models exist wherein higher CTR translates into better remuneration for ad agencies. As such, models that can predict the CTR of a given set of parameters can help ad agencies choose between alternative proposals, minimizing the need for audience test strategies, reducing workloads, associated costs, and lead times, and ultimately improve profit margins.

On a related note, a rising trend among advertisers is the development and use of generative text models and systems that can automate the production of text for use in ads, whether as the primary content or as a component in other ad formats [1]. Some such systems have been released for use by individuals or small business owners, who do not necessarily have the expertise to know what to write, nor the experience to know what works best. CTR prediction models coupled to such generative systems can lead to highly-optimized workflows and results.

That is not to say that human knowledge and experience is no longer useful. Though CTR prediction is generally not a part of the traditional ad creation process, ad creators have always sought to maximize CTR through their choices and craft. It is my belief that this somewhat nebulous body of knowledge can be leveraged to produce a system that lends clarity and form to the ad creation process better than black-box models.

1.2 Clarification

Much of the work done on CTR prediction in the digital ad space appears to be focused on the task of predicting whether an ad under a given set of circumstances is going to be clicked or not. Many popular datasets used for CTR prediction for online ads (Criteo and Avazu datasets) present this task, with various metadata variables and anonymized categorical values, the semantics of most of which are not publicly known [2, 3].

There appears to be very little publicly-available work on CTR prediction using the actual text of the ads. This is not particularly surprising, as ad agencies and platforms are generally not at liberty to release the texts alongside performance metrics. Some of the categorical variables in the aforementioned datasets may actually be representations of the ad copy texts, but most researchers still do not have direct access to the plaintext corpora of particular interest in the primary task that was the focus of this study.

1.3 Task Definition

The task in question was to predict the CTR of sets of texts (described in more detail in a later section) intended for use in text ads, specifically those served up in relation to a search query on a search engine.

As I touched upon in the section on motivations, one of the intentions behind the development of this system was to support our ad creative teams in finding the best phrasing for ad copy texts that would lead to maximum profits. Of course, such a system could also be integrated with a generative text model to rank the generated texts, providing some insight as to which texts would provide the best results.

To that end, we were more interested in the effects of changes in the texts themselves, rather than how they would perform in specific circumstances, as could be described by the variables representing the context surrounding the ad (e.g. (Avazu) site_category, app_domain) and the user in question (e.g. (Avazu) device_id, device_ip).

On a related note, considering the advent of the “cookie-less Internet” [4], approaches that rely on access to third-party cookies to identify the context and user in question may be less viable, or at least will become more challenging. This could mean a shift away from strategies that target specific user segments or even specific users, and back towards focusing efforts on the ads’ contents to maximize clicks. This was a consideration that partly informed our decision to focus on the primary task as described.

2 Our Approach

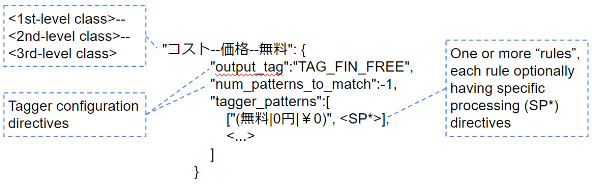
2.1 Handcrafted Feature Extraction

Of central interest to this study was the development and testing of a feature extractor that used input from our creative teams, with regards to what they thought were factors that affected the performance of our ads.

In a preceding internal project, we had the goal of identifying factors that were adjustable or controllable within the ad creation process; the idea being that if we could analyze ads and metrics such that we could tell which elements contributed to better CTR, our creative teams could adjust their future work to incorporate more of those elements. Conversely, elements that actually tended to decrease CTR were also identified, and were to be avoided whenever possible in future work.

A subset of this previous work was directly related to the text used in banner ads. Although not expressly developed for the analysis of search ads, we deemed there was sufficient overlap that the schema could be used to explicitly extract features that could be used to gain insight into search ads as well.

The feature extractor developed using this tag schema was built on aggregations of regular expressions. When a regular expression representing certain words or phrases turned up a match, a tag was assigned to the text in question.


<1st-level class>--
<2nd-level class>--
<3rd-level class>
Tagger configuration directives
One or more “rules”, each rule optionally having specific processing (SP*) directives

```
"コスト--価格--無料": {
        "output_tag":"TAG_FIN_FREE",
        "num_patterns_to_match":-1,
        "tagger_patterns":[
            ["(無料|0円|￥0)", <SP*>],
            <...>
        ]
    }
```

Fig. 1 Example of configuration for a tagger

For any given singular ad, there are multiple text fields, so each field was tagged separately, such that the resulting handcrafted feature (hereafter referred to as *“tag-based feature(s)”*) column count for a given expanded search ad was 866 * 6 = 5,196. However, this was a very large number of sparse dimensions, so it was reduced by aggregating tags by merging the features of the 3 ad title fields together and those of the 2 description fields together, such that the resulting number of tag-based feature columns was 866 * 2 = 1732. The only exception to this is the LightGBM experiment detailed below, as we ended up applying a different means of reducing dimensions in that case.

Metadata features were also converted into n-hot encodings. This amounted to a column count of 964, a majority of which were representations of the account IDs associated with the ads.

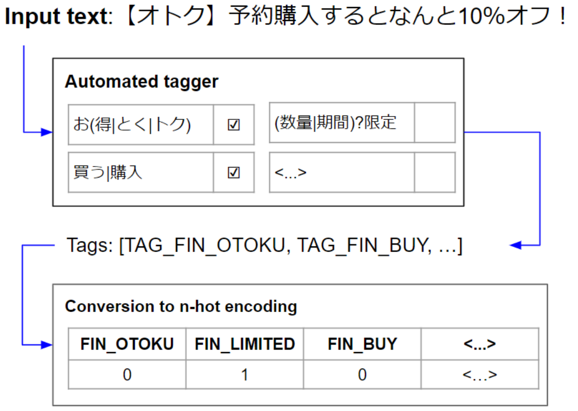

Input text:【オトク】予約購入するとなんと10%オフ！
Automated tagger
お(得|とく|トク) ☑
(数量|期間)?限定
買う|購入 ☑
<...>
Tags: [TAG_FIN_OTOKU, TAG_FIN_BUY, …]
Conversion to n-hot encoding
FIN_OTOKU
FIN_LIMITED
FIN_BUY
<...>
0
1
0
<...>

Fig. 2 Handcrafted feature extraction

These features were extracted and saved to disk, to reduce the amount of computation and time needed during training.

Although not explicitly examined in this study, the handcrafted features are also directly translatable into outputs that can inform end-users as to what textual features affect the predicted CTR.

3 Experiments

We performed experiments in several different paradigms. These were all performed using scikit-learn or PyTorch on Google Colab instances, generally with a maximum of 12GB RAM and 16GB VRAM. Input tensors used in training and validation were also cached to disk after generation during the first epoch to reduce processing time.

3.1 Data

We used data collected using our company’s ad performance tracking systems. Specifically, we used the metadata, text fields, impression and click counts for Japanese language expanded text ads on Google, which were active between 2017/07/01 and 2021/02/24 (the time of extraction), compiled by month, grouped separately by ad ID and device type. The individual ad IDs (as opposed to ad account IDs) were not factored into learning. We also filtered out instances with fewer than 100 monthly impressions.

Table 1. Data fields used

Metadata	Ad account ID, year and month, device type, network type,
TD Fields	Ad title 1, ad title 2, ad title 3, description 1, description 2

The functions and limitations of the individual TD (title and description) fields are explained here: [5].

The total row count was 1,453,181. Train-validate-test splitting was performed with ratios of 0.7, 0.2, and 0.1 resulting in 1,017,227 training rows, 290,637 validation rows, and 145,317 test rows.

3.2 LightGBM

Using the n-hot encoded tag-based features did not show very promising results in the short time we devoted to this confirmatory experiment. Instead, we converted the permutations of the n-hot tag-based features *per TD field* and assigned integer values to be used as categorical value indices. We then performed a K-fold (n_splits=5, shuffle=True) training regime using the metadata fields as categorical features and the categorically-encoded feature fields.

3.3 Multilayer Perceptron ("MLP")

After experimenting with various network shapes (partly informed by [6]), layer depths, and node counts, we achieved the results presented in the next section with a network with 5 hidden layers of 600 nodes each. The input used n-hot encoded metadata fields (964 columns) and n-hot encoded TD feature fields (866 + 866); the activation function was ReLU.

3.4 Fine-tuned DistilBERT

We decided to perform our experiments using DistilBERT-base-japanese [7]. We selected DistilBERT because it has been shown to perform only slightly worse than a comparable BERT model, while reducing memory and processing requirements by more than three-quarters, which was in line with one of our end-goals of producing an interactive web-based tool.

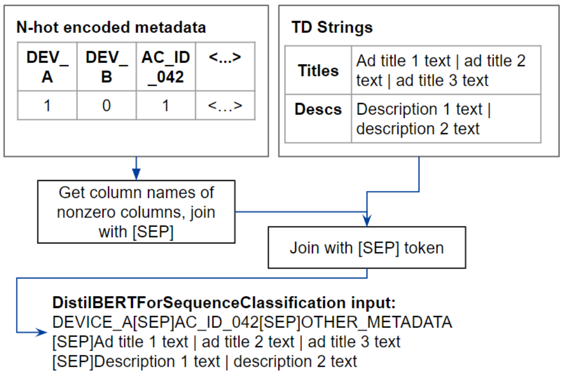

N-hot encoded metadata
DEV_A
DEV_B
AC_ID_042
<...>
1
0
1
<...>
TD Strings
Titles
Ad title 1 text | ad title 2 text | ad title 3 text
Descs
Description 1 text | description 2 text
Get column names of nonzero columns, join with [SEP]
Join with [SEP] token
DistilBERTForSequenceClassification input:
DEVICE_A[SEP]AC_ID_042[SEP]OTHER_METADATA
[SEP]Ad title 1 text | ad title 2 text | ad title 3 text
[SEP]Description 1 text | description 2 text

Fig. 3 Including metadata features as text for DistilBERT

We took the DistilBERT-base-japanese model publicly available via HuggingFace, and fine-tuned it using the DistilBertForSequenceClassification [8] class with num_labels=1. The input for this model was expected to be text strings, so I converted the existing n-hot metadata encodings using the column headings, joined them with [SEP] tokens, and prepended them to the composite ad title and composite description strings, also separated with [SEP] tokens.

3.5 Ensemble

A very simple ensemble model that took the results of predictions from two disparate models and presented the mean of those values as its prediction was also developed. The internal models were an instance of the aforementioned MLP models and a fine-tuned DistilBERT model. The input for the MLP portion was n-hot encoded metadata and tag-based features. The DistilBERT portion was provided with the TD strings joined by [SEP] tokens.

3.6 Joint Learning

Finally, taking cues from [9], we constructed a joint learning network using discrete subnetworks for each modality of data, with another network built on top, using the concatenated hidden states of the subnetworks to perform the regression task.

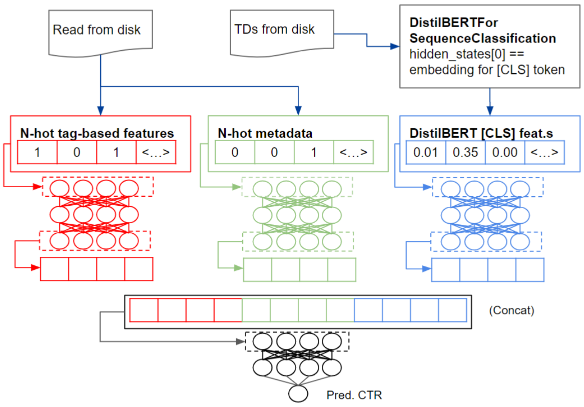


Fig. 4 Diagram of Joint Learning Model

The figure above shows a joint learning model using 3 subnetworks, one for each modality of data. The three subnetworks and the regressor network (depicted after the concat step) all use MLP-like networks with dropout.

4 Results

The results presented in Table 2 are based on tests using the same test data, with the exception of the LightGBM experiment which was evaluated with K-fold cross-evaluation. The ground truth values for CTR are in decimal form (0.0 ~ 1.0), and the predicted values are normalized to the same scale before computing the error values.

The results in Table 3 describe the percentages of test cases that could be classified under the following categories:

- Overshoot – predicted value was greater than the ground truth value by more than 0.05.

- Undershoot – predicted value was less than the ground truth value by more than 0.05.
- Acc. (tol .05) – the absolute difference between the predicted and ground truth values was less than or equal to 0.05. *This includes values categorized as VAcc.*
- VAcc. (tol .005) – the absolute difference between the predicted and ground truth values was less than or equal to 0.005.

Table 2. Primary results comparison

	Corr.	MAELoss	RMSELoss
LightGBM	0.7586	0.03000	0.05458
MLP	0.7547	**0.02933**	0.05428
Fine-tuned DistilBERT	0.7188	0.03543	0.05866
Ensemble	**0.786**3	0.03192	**0.05065**
Joint Learning	0.7660	0.03240	0.05257

Table 3. Secondary results comparison

	Over-shoot	Under-shoot	Acc. (tol .05)	VAcc. (tol .005)
LightGBM	**4.09%**	12.37%	83.54%	**23.49%**
MLP	4.71%	10.90%	**84.49%**	23.23%
Fine-tuned DistilBERT	12.31%	8.79%	78.90%	15.95%
Ensemble	10.47%	**8.27%**	81.26%	15.14%
Joint Learning	9.73%	9.06%	81.21%	16.47%

※Corresponding graphs included in the Appendix.

4.1 Discussion

Our intention with the LightGBM experiment was to provide some baseline for the level of accuracy we can expect using our features. Though it performed well, the reduction of dimensions through categorical encoding of the permutations of the features leaves some doubt as to how it will behave with out-of-distribution patterns.

Based on the coefficient of correlation, the ensemble model performs the best. However, if we consider the distribution of accurate predictions, it is edged out by the MLP approach.

On a side note, we also tried an ensemble model where the DistilBERT submodel was finetuned with both metadata and TD strings – the very same model that performed best in our Fine-tuned DistilBERT experiments. However, it ended up bringing the performance of the ensemble model down significantly (corr. 0.6873, MAELoss 0.03935, RMSELoss: 0.06284).

The apparent non-linear relationship between coefficient of correlation and loss values (also evidenced by differences in Table 3) may be attributed to the inherently lopsided distribution of our data. Almost 93% of our test data had ground truth CTRs of less than or equal to 0.2 (20%), and the models that tended to do well in this range tended to do less well with instances where the ground truth CTR was higher, leading to lower overall coefficients of correlation.

4.2 Further Research

Note that the experiments as described above are the best results we achieved thus far for their respective paradigms. In particular, we feel that more experimentation can be performed with joint learning to achieve better results.

Additionally, I also feel that there must be other network structures that are more befitting of the dense, continuous values of the DistilBERT-based (and other BERT-based) features. So far, I've implemented MLP-like networks to make comparison with the other paradigms relatively simple, but considering the fundamental differences in the kind of features involved, I feel that greater expressivity can be preserved through the use of alternative activation functions and node distributions.

Additional work on different combinations of modalities in the ensemble and joint learning approaches may also lead to more optimal configurations.

Comparative experiments using embeddings from n-gram, skip-gram and CBOW feature generators may also be useful in examining the power of our approach, although systems using such may be more computationally intensive due to the curse of dimensionality.

Some insight may also be gained by comparing the computational costs and requirements of the different approaches, especially since we intend to deploy them in real-time applications where responsiveness is a strong requirement.

Finally, we are preparing to see if the same approach can be transferred to other ad formats, especially considering that the ad format that our data was in is about to be sunsetted.

4.3 Conclusion

In this paper we have shown that a sufficiently comprehensive set of simple handcrafted features can be used to parameterize text ads for the purpose of predicting click-through rates.

Additionally, the handcrafted features are directly translatable into human-readable labels, providing greater transparency and explainability than when using generalized language models like DistilBERT. This explainability can be particularly useful in providing feedback for end-users who use our system to augment and accelerate their creative workflows.

Acknowledgements

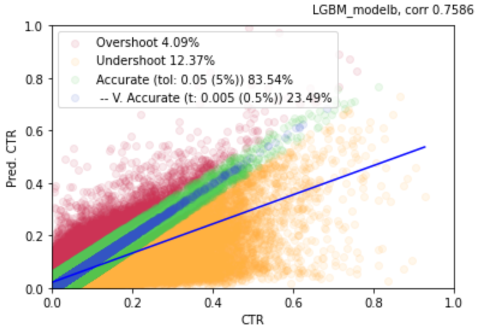
I would like to thank my colleagues at Opt, Inc. for their feedback and support; especially to the members involved in a previous project that resulted in the creation of the tag schema I used as the basis of this entire endeavor. I would also to thank, specifically:

- 田中宏明 Tanaka Hiroaki, my manager, for keeping me on track and general support.
- 粟飯原俊介 Aihara Shunsuke, for providing guidance and insights throughout the entire project.
- 岩田大地 Iwata Daichi, who helped me directly by performing EDA and implementing the LightGBM experiment.

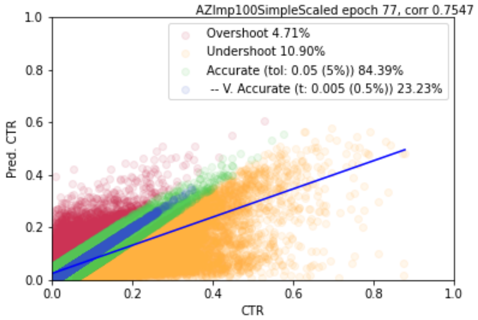
References

1. Automatic Generation of Title and Description Texts for Sponsored Search Ads. **Baba Jun, Iwazaki Yuki, Itsuki Sugio, Kitade Kosuke, and Fukushima Takeshi**. The 29th Annual Conference of the Japanese Society for Artificial Intelligence, 2015.
2. Data description of Criteo dataset. (Online) (Last accessed 2021-01-07). https://www.kaggle.com/c/criteo-display-ad-challenge/data
3. Data description of Avazu dataset. (Online) (Last accessed 2021-01-07). https://www.kaggle.com/c/avazu-ctr-prediction/data.
4. Digital Marketing In A Cookie-Less Internet. **Juneau Todd**. (Online) (Last accessed 2022-01-07). https://www.forbes.com/sites/forbesagencycouncil/2020/05/18/digital-marketing-in-a-cookie-less-internet/?sh=d323da721e2d
5. About expanded text ads. (Online) (Available as of 2021-12-16). https://support.google.com/google-ads/answer/7056544?hl=en
6. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. **Guo Huifeng, Tang Ruiming Tang, Ye Yunming, Li Zhenguo, He Xiuqiang**. Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, 2017, 1725-1731.
7. BANDAI NAMCO Research Inc. DistilBERT-base-jp on GitHub (Online) (Last accessed 2022-01-07). https://github.com/BandaiNamcoResearchInc/DistilBERT-base-jp
8. —. DistilBERT on HuggingFace. (Online) (Last accessed 2022-01-07). https://huggingface.co/docs/transformers/model_doc/distilbert#transformers.DistilBertForSequenceClassification
9. Click-Through Rate Prediction of Online Banners Featuring Multimodal Analysis. **Xia Bohui, Seshime Hiroyuki, Wang Xueting, and Yamasaki Toshihiko**. International Journal of Semantic Computing, 2020, Vol. 14 No. 1, 71-91.

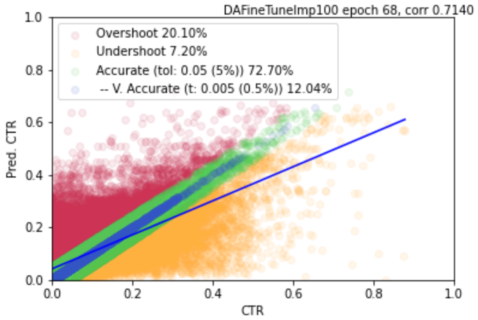
Appendix



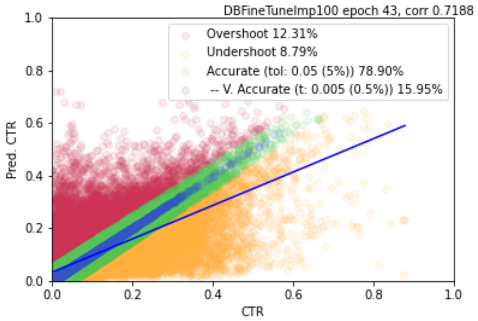
Plot of LightGBM model test



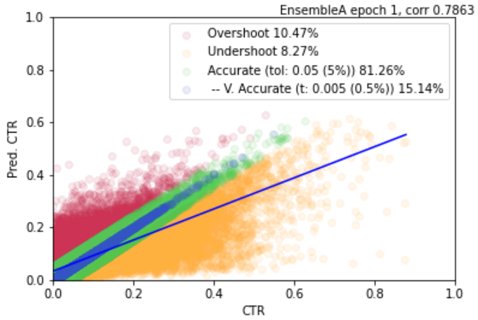
Plot of MLP model test



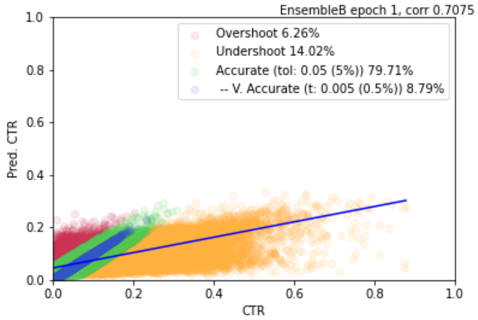
Plot of Fine-tuned DistilBERT model test, trained with TD texts only



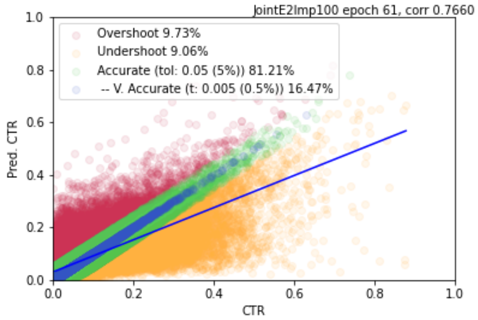
Plot of Fine-tuned DistilBERT model test, trained with metadata and TD texts



Plot of Ensemble model test, using DistilBERTForSequenceClassification fine-tuned with TD texts only



Plot of Ensemble model test, using DistilBERTForSequenceClassification fine-tuned with both metadata and TD texts.



Plot of Joint Learning model test