Exploiting Contextual Normalizations and Article Endorsement for News Recommendation

Exploiting Contextual Normalizations and Article Endorsement for News Recommendation

Andrea Alari
Politecnico di Milano
Milano, Italy
andrea.alari@mail.polimi.it

Lorenzo Campana
Politecnico di Milano
Milano, Italy
lorenzo.campana@mail.polimi.it

Federico Giuseppe Ciliberto
Politecnico di Milano
Milano, Italy
federicogiuseppe.ciliberto@mail.polimi.it

Saverio Maggese
Politecnico di Milano
Milano, Italy
saverio.maggese@mail.polimi.it

Carlo Sgaravatti
Politecnico di Milano
Milano, Italy
carlo.sgaravatti@mail.polimi.it

Francesco Zanella
Politecnico di Milano
Milano, Italy
francesco2.zanella@mail.polimi.it

Andrea Pisani
Politecnico di Milano
Milano, Italy
Politecnico di Torino
Torino, Italy
andrea.pisani@polito.it

Maurizio Ferrari Dacrema
Politecnico di Milano
Milano, Italy
maurizio.ferrari@polimi.it

Abstract

We provide an overview of the approach used as team FeatureSalad for the ACM RecSys Challenge 2024, organized by Ekstra Bladet. The competition addressed the problem of News Recommendation, where the goal is to predict which article a user will click on given the list of articles that are shown to them. Our solution is based on a stacking ensemble of consolidated algorithms, such as gradient boosting for decision trees and neural networks. It relies on numerous features, which model the interest of a user and the lifecycle of an article. The proposed solution allowed our team to rank first among the academic teams, and sixth overall.

CCS Concepts

• **Computing methodologies** → **Learning to rank**; **Supervised learning by classification**; **Classification and regression trees**; **Natural language processing**.

Keywords

ACM Recsys Challenge 2024, Gradient Boosting for Decision Trees, Neural Networks, News Recommendation, Recommender Systems, Stacking

ACM Reference Format:
Andrea Alari, Lorenzo Campana, Federico Giuseppe Ciliberto, Saverio Maggese, Carlo Sgaravatti, Francesco Zanella, Andrea Pisani, and Maurizio Ferrari Dacrema. 2024. Exploiting Contextual Normalizations and Article Endorsement for News Recommendation. In *ACM RecSys Challenge 2024 (RecSys Challenge '24), October 14–18, 2024, Bari, Italy.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3687151.3687154

RecSys Challenge '24, October 14–18, 2024, Bari, Italy

ACM ISBN 979-8-4007-1127-5/24/10
https://doi.org/10.1145/3687151.3687154

1 Introduction

Numerous online platforms make use of Recommendation Systems to assist users in finding content that aligns to their preferences. News platforms are no exception: accurate news recommendations have a significant impact on user experience [21]. The ACM RecSys Challenge 2024[1], organized by Ekstra Bladet, aims to identify the best approach to predict the likelihood of user clicks on news articles within a given impression. This prediction task comes with several hurdles, e.g. handling a large-scale dataset, and evaluating accuracy with limited computational resources. We propose an optimized stacked ensemble model, able to successfully leverage a wide and diverse feature set. The proposed solution accurately predicts user behaviors, which allowed our team to lead the academic leaderboard and place sixth overall. The source code of our final model and its documentation are publicly available on GitHub [2].

2 Problem Formulation

The dataset for the ACM RecSys Challenge 2024, named EB-NeRD, is a large-scale Danish dataset created by Ekstra Bladet to support advancements and benchmarking in News Recommendation research. EB-NeRD includes data from over 2.3 million users and more than 380 million impression logs collected from Ekstra Bladet. It was compiled by recording *behavior logs* from active users during a six-week period from April 27 to June 8, 2023. This specific timeframe was chosen to avoid major events, such as holidays or elections, that could result in atypical user behavior on Ekstra Bladet. To protect user privacy, the logs were anonymized using one-time salt mapping. In addition to user interaction data, the dataset includes news articles published by Ekstra Bladet, enriched

[1] http://www.recsyschallenge.com/2024/
[2] https://github.com/recsyspolimi/recsys-challenge-2024-ekstrabladet

with textual content features such as titles, abstracts, bodies, and categories. Moreover, the dataset provides features generated by proprietary models, including topics, named entity recognition (NER), and article embeddings. The Challenge's objective is to estimate the likelihood of a user clicking on any candidate article by evaluating the compatibility between the article's content and the user's preferences. The articles are ranked based on these likelihood estimations, and the accuracy of these rankings is measured against the actual selections made by users. Solutions are evaluated using the Area Under the receiver operating characteristic Curve (AUC).

3 Data Analysis

Three versions of the dataset, *demo*, *small*, and *large*, were provided, each split into self-contained training and validation sets. Every set included *behavior logs* and *history logs*. *Behavior logs* contained records of user impressions collected during a seven-day period. An impression consists of the randomly shuffled list of articles a given user was shown, called *inview list*, and a list of the ones they clicked on. The *inview lists* contain a minimum of 5 and a maximum of 100 articles. *History logs* provide a list of the articles a given user has clicked on over the 21 days prior to the recording of their *behavior logs*. Many user profiling features such as age, gender, and postcode had missing values. The impressions in the dataset exhibit both weekly and daily seasonality, i.e. the distribution of impressions depends on the considered day of the week and the time of day. For instance, we have found the majority of impressions to happen around 6 a.m. Regarding the articles, a strong preference is noticeable for those pertaining to the "Kendt" (celebrities) topic.

4 Feature Engineering

In the following Section, we describe the most relevant features we devised for our solution.

4.1 Temporal and Contextual Features

Temporal and contextual features aim to model how the timing and popularity of an article or topic are connected to a user's click probability.

4.1.1 Trendiness at n. Each article a in the *inview list* was scored according to how trendy its topics $\mathcal{T}(a)$ were when the impression happened. This is computed as the sum over all its topics of the times that an article with that topic has been published in the n days preceding the impression time. The goal was to measure the extent to which an impression aligns with the most discussed topics of the moment. We also compare *Trendiness* across multiple periods, by calculating the ratio between *Trendiness* scores with different values of n (e.g. 1 and 3). Furthermore, we exploit *history logs* to compute the *Mean Topic Trendiness*, i.e. the mean *Trendiness* score over all topics of article a:

$$MTT(a) = \frac{1}{|\mathcal{T}(a)|} \sum_{t \in \mathcal{T}(a)} \frac{1}{|I(t)|} \sum_{i \in I(t)} T(a_i). \quad (1)$$

Note that $I(t)$ is the set of interactions in the history with articles containing topic t, a_i is the article that was clicked within impression i, and $T(a_i)$ is its *Trendiness*.

4.1.2 Endorsement. We define *Endorsement* as the number of appearances of article a in the *inview list* of any user, in the last m hours before the considered impression timestamp. We also define *Endorsement Article-User* as a count of how many times article a appeared in the same user's *inview list* in previous impressions in the last l hours.

4.1.3 Article Delay. To represent how recent an article is, we calculate the distance in days and hours between the considered impression's timestamp and the article's publication time. We also use *history logs* to compute *Mean Topic Delay*, as described in equation 1 – substituting *Article Delay* to the *Trendiness* operator T.

4.1.4 Leaking Features. Recently published articles produce poorly significant results for features such as *Endorsement* and *Trendiness*. To accurately score them, knowing future trends and behaviors is essential. Following this reasoning, we compute a version of said features that accounts for future impressions. Furthermore, we also classify as *Leaking Features* some of those present in the raw dataset, such as *Total Pageviews*, which account for future behaviors with respect to the considered impression. In Section 6.2 we evaluate the impact of *Leaking Features*.

4.2 User History Features

4.2.1 Article Topics Jaccard Similarity. It is possible to compute a topics-based similarity between two articles as the Jaccard Similarity between their topic sets. Given a user's history h and an article a in their *inview list*, we compute such similarity between a and all the news in h. Various aggregations (*mean*, *max*, *std*, *min*, ...) are finally computed over this list of similarities and used as features.

4.2.2 Topics TF-IDF Cosine Similarity. An alternative formulation for the similarity between a user's history and a given article a. A TF-IDF vectorizer is fit on the topics of all the dataset's articles. Such vectorizer allows to embed a's list of topics in a vector. The user's history can be similarly embedded by concatenating all the topics in all its articles and processing it through the TF-IDF vectorizer. Finally, the cosine similarity between the two embedding vectors is computed and used as a feature.

4.2.3 Latent Dirichlet Allocation. A third similarity formulation to compare a candidate article to a given user's history. We fit a 5-dimensional Latent Dirichlet Allocation (LDA) model on the articles' titles. This model is then used to embed user histories and candidate articles, and compare them using cosine similarity. The similarities are then aggregated as in 4.2.1. Additionally, we compute the mean LDA embedding for each user's history and treat it as 5 features.

4.2.4 Mean User Trendiness. Users tend to show different levels of interest in topics that are currently popular. We model such behavior by averaging the *Trendiness* of the interactions in the users' history.

4.2.5 Mean User Delay. The mean, computed over all the articles in a user's history, of the time distance between the click timestamp and the article's publication time.

4.2.6 User Behavior Statistics. We computed aggregations over user histories regarding the contained articles' *reading time* and

scroll percentage features. Additionally, *User Behavior Statistics* include the frequency of each article category in a user's history. Notably, we found said frequency to be relevant only for a few categories (e.g. "Sport").

4.3 Embeddings and ICM Features

Four artefacts containing article embeddings were provided. They were obtained using four different models, having each article's title, subtitle, and body as input. We computed new embeddings of article titles and subtitles, using DistilBERT [17] and a Danish version of MiniLM [19], both available on the HuggingFace hub. They were created to give more relevance to titles and subtitles, considering that users only see those elements from the user interface. Each embedding set was used to build an Item Content Matrix (ICM), having the articles' IDs as rows and a column for each dimension of the embedding. Each ICM was then paired with the implicit User Rating Matrix (URM) and used as input for an ItemKNN model [8], exploiting the artefacts to improve its accuracy w.r.t pure Collaborative Filtering. The scores produced by the ItemKNNs were then used as features.

4.4 Feature Normalizations and Aggregations

To enable a clear comparison of some feature values for articles contained in the same *inview list*, we construct an additional set of derived features, consisting of a series of normalizations. Each selected feature is grouped in three different ways (by *impression ID*, by *user ID*, and by *article ID*) before being normalized, thus generating three new features per applied normalization. This process also helps in mitigating the great variance that unnormalized features may show.

4.4.1 Max Normalization. Each feature's value is normalized by the maximum over the selected group of rows, aiming at having comparable feature ranges among different groups. If the features are positive, this corresponds to a l_∞ normalization.

4.4.2 Median Subtraction. To represent how much a feature value for an article differs from other articles in the selected row group, we subtract the median group value from the feature's raw value. We use the median since it is more robust to outliers than the mean.

4.4.3 Ranking. We rank each article based on the value of the considered feature in descending or ascending order, based on which order is more relevant (e.g. descending for *Endorsement* and ascending for *Article Delay*).

4.4.4 Feature Aggregations. We provided other contextual information to our models by computing aggregations of feature values over their *impression ID*, such as the standard deviation, skew, kurtosis and entropy. The aim of said aggregations is to provide a representation of the feature value distribution inside the impression's *inview list*. Furthermore, to exploit the temporal properties of the *Endorsement* feature, we have processed it in the following ways:

Temporal Sum Normalization: we divide by the sum of the *Endorsement* of all the articles on the same m-hour time period. The resulting feature, which is an l_1 normalization, can be interpreted as the percentage of user u's impressions in the last m hours which contained article a in the *inview list*.

Temporal Quantile Normalization: we divide by the α-quantile of the *Endorsement* value distribution over the articles on the same time period, having $\alpha = 0.8$. We use the 80th percentile instead of the maximum to remove outlier values.

Rolling Average Subtraction: given an article a's *Endorsement*, we subtract its rolling average over a time window of length w_1, to capture temporal trends. Similarly, we include an additional feature, using another rolling average of window $w_2 > w_1$ in place of the raw *Endorsement* value.

4.5 Feature Selection

Our complete dataset originally had 392 features. Given its size, it was beneficial to create a streamlined version by retaining only the features that significantly impacted the model's AUC. To achieve this, we performed feature selection using the null importances method [2], resulting in the removal of 136 features. The reduced dataset thus obtained was employed for training particularly resource-intensive models and for the second Tier of the stacking model.

5 Models

We have experimented with two different classes of models: Gradient Boosting Trees (GBTs) and Neural Networks. We modeled the problem both as a classification task, predicting the click probability of a candidate article, and as a ranking task, predicting the article's relevance inside the *inview list*.

GBTs are the state-of-the-art [10] solutions for tabular datasets in terms of performance and training time [3, 5, 7, 9, 12]. In our experiments, we tested *Catboost* [16] and *LightGBM* [14], using both their ranking and classification versions. Both classification versions were trained using the logloss function, while the rankers followed different algorithms (YetiRank [11] and LambdaRank [4], respectively).

The neural networks were trained for classification, using the binary cross-entropy loss function. For all of them, we preprocess the feature distributions using the Yeo-Johnson transformation [22]. We implemented a *Multi-Layer Perceptron* [15], a *GANDALF* network [13], a *Deep&Cross* network [18], and a *Wide&Deep* one [6]. The code for each of said implementations is provided on our public GitHub repository.

5.1 Ensemble

We merge all the previously described models into a 2-tier stacking ensemble [20]. The training phase works as follows: all Tier 1 models are trained on the training set and used in inference mode over the validation set; the obtainied predictions are normalized with respect to *impression*, *article*, and *user ID*; some relevant metrics such as the rank position per impression are also extracted. These new features are combined with the selected features described in Section 4.5 in order to build the dataset for the Tier 2 model. For Tier 2, a LightGBM classifier and a Catboost classifier were experimented, with Catboost being the one chosen for our best submission. After tuning the hyperparameters, Tier 1 models are finally retrained on the concatenation of train and validation set.

Table 1: Performance of our models on the test set.

Model	AUC	MRR
Catboost Ranker	0.8422	0.6522
Catboost Classifier	0.8362	0.6401
LightGBM Ranker	0.8356	0.6385
LightGBM Classifier	0.8346	0.6414
MLP	0.8287	0.6295
GANDALF	0.8249	0.6232
Deep&Cross	0.8265	0.6228
Wide&Deep	0.8212	0.6168
Catboost Classifier L2	0.8513	0.6658
LightGBM Classifier L2	0.8498	0.6635

The obtained predictions are used as previously described to build the dataset for Tier 2.

5.2 Hyperparameter Tuning

We performed hyperparameter tuning for all our models on the *small* dataset due to limited computational resources. We deemed this not to be a problem due to the strong correlation between our test outcomes on the *small* and *large* datasets. AUC was chosen as target metric to optimize. We exploit Bayesian Optimization for all our models, using the Optuna framework [1]. Tier 1 models are tuned using the provided splits. The ItemKNN models described in Section 4.3 were also tuned as Tier 1 models during the data preprocessing stage. Tier 2 models are tuned using an additional split of the training set's *behavior logs*: the optimized Tier 1 models were trained over the first 3.5 days to compute predictions over the remaining days, which were then used to optimize Tier 2.

6 Results

Table 1 shows the AUC and Mean Reciprocal Rank (MRR) results achieved by all our models on the official test set. Catboost is the best performing Tier 1 model. We can argue its leading position is due to the presence of mixed features in our dataset, many of which categorical. In general, GBT models provide better accuracies than neural networks. Nonetheless, the neural network models were still important in our final ensemble model, as they provided diverse recommendations with respect to the GBTs. The final ensemble model brought to a 1.08% increase in the AUC score, proving the effectiveness of combining the predictions of our models. Our best AUC score was 0.8513.

6.1 Feature Importance

Figure 1 shows the importance of our features for the Tier 1 Catboost Classifier model. *Endorsement* and *Endorsement Article-User* were the most important features; furthermore, it is interesting to notice how normalized *Endorsement* versions were more important than the actual raw value, validating the reasoning described in Section 4.4. Moreover, *Article Delay* was important for our models, ranking very high in its various forms. The ratio between *Leaking Features* such as *Total Pageviews* and *Total Inviews* was the seventh most important feature. Finally, the features based on content,

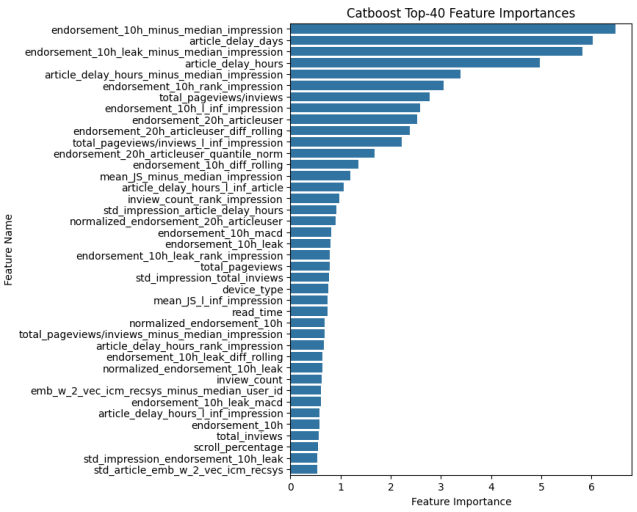

Figure 1: The importance of the top-40 features for the Tier 1 Catboost Classifier model.

Table 2: Impact of leaking features on the small validation set.

Model	AUC (w/ leaks)	AUC (w/o leaks)
Catboost Classifier	0.8165	0.7949
Catboost Ranker	0.8182	0.8020
LightGBM Classifier	0.8166	0.7946
LightGBM Ranker	0.8233	0.8010
MLP	0.8074	0.7846
GANDALF	0.8056	0.7826
Wide&Deep	0.8092	0.7868
Deep&Cross	0.7934	0.7745

such as the *Article Topics Jaccard Similarity* and the ICM features, respectively at positions 13 and 34, show relative importance as well. Overall, the *Median Subtraction* and the *Max Normalization* were the most effective normalizations, together with the standard deviation aggregations.

6.2 Ablation Studies

Leaking Features in our dataset account for information about future impressions, so they would not be available in a live setup. To assess the quality of our solution in a live setup, *Leaking Features* have been obscured and several experiments have been conducted, using the *small* dataset. The results are shown in Table 2. We can conclude that the *Leaking Features* increased AUC by an average of 2.68%.

We performed an additional ablation study on the Tier 1 Catboost Classifier model to assess the impact of several components of our project. We tested a baseline model on the *small* dataset, then added the feature normalizations discussed in Section 4.4, then performed a final test over the full *large* dataset, i.e., the combined training set and validation set. Said study showed that integrating feature normalizations to our initial raw features on the official test set

amounted to an AUC increase of 0.0809 (+11.02%), and training on the full dataset has brought an additional AUC improvement of 0.0209 (+2.56%).

7 Conclusions

The ACM RecSys Challenge 2024 aimed at predicting the probability of clicking candidate articles within a given impression in a News Recommendation scenario. Our solution combines information captured by the features we extract from the dataset regarding articles, histories, and behaviors. Said features allow us to model the users' behavior and article trends within various time scopes. It integrates predictions from different Gradient Boosting Trees (GBT) and neural network models. A boost in performance was given by using as features the scores of hybrid recommenders that exploited article embeddings as content. Normalizing features over the impressions played an important role in reaching our best results, as we found that contextual information was fundamental in predicting the next clicked article. The stacking ensemble technique we used provided an improvement with respect to the performance of the single models in both local and leaderboard evaluations, hence we were capable of reaching the first position among academic teams at the end of the competition, and the sixth position in the overall standings.

Acknowledgments

We would like to thank Prof. Paolo Cremonesi for his support.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[2] André Altmann, Laura Tolosi, Oliver Sander, and Thomas Lengauer. 2010. Permutation importance: a corrected feature importance measure. *Bioinform.* 26, 10 (2010), 1340–1347. https://doi.org/10.1093/BIOINFORMATICS/BTQ134

[3] Paolo Basso, Arturo Benedetti, Nicola Cecere, Alessandro Maranelli, Salvatore Marragony, Samuele Peri, Andrea Riboni, Alessandro Verosimile, Davide Zanutto, and Maurizio Ferrari Dacrema. 2023. Pessimistic Rescaling and Distribution Shift of Boosting Models for Impression-Aware Online Advertising Recommendation. In *ACM RecSys Challenge 2023, Singapore, 19 September 2023*. ACM, 33–38. https://doi.org/10.1145/3626221.3627288

[4] Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. 2006. Learning to Rank with Nonsmooth Cost Functions. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, Bernhard Schölkopf, John C. Platt, and Thomas Hofmann (Eds.). MIT Press, 193–200. https://proceedings.neurips.cc/paper/2006/hash/af44c4c56f385c43f2529f9b1b018f6a-Abstract.html

[5] Luca Carminati, Giacomo Lodigiani, Pietro Maldini, Samuele Meta, Stiven Metaj, Arcangelo Pisa, Alessandro Sanvito, Mattia Surricchio, Fernando Benjamín Pérez Maurera, Cesare Bernardis, and Maurizio Ferrari Dacrema. 2021. Lightweight and Scalable Model for Tweet Engagements Predictions in a Resource-constrained Environment. In *RecSys Challenge 2021: Proceedings of the Recommender Systems Challenge 2021, Amsterdam, The Netherlands, 1 October 2021*. ACM, 28–33. https://doi.org/10.1145/3487572.3487597

[6] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016*, Alexandros Karatzoglou, Balázs Hidasi, Domonkos Tikk, Oren Sar Shalom, Haggai Roitman, Bracha Shapira, and Lior Rokach (Eds.). ACM, 7–10. https://doi.org/10.1145/2988450.2988454

[7] Nicola Della Volpe, Lorenzo Mainetti, Alessio Martignetti, Andrea Menta, Riccardo Pala, Giacomo Polvanesi, Francesco Sammarco, Fernando Benjamin Perez Maurera, Cesare Bernardis, and Maurizio Ferrari Dacrema. 2022. Lightweight Model for Session-Based Recommender Systems with Seasonality Information in the Fashion Domain. In *Proceedings of the Recommender Systems Challenge 2022* (Seattle, WA, USA) *(RecSysChallenge '22)*. Association for Computing Machinery, New York, NY, USA, 18–23. https://doi.org/10.1145/3556702.3556829

[8] Mukund Deshpande and George Karypis. 2004. Item-based top-N recommendation algorithms. *ACM Trans. Inf. Syst.* 22, 1 (jan 2004), 143–177. https://doi.org/10.1145/963770.963776

[9] Nicolò Felicioni, Andrea Donati, Luca Conterio, Luca Bartoccioni, Davide Yi Xian Hu, Cesare Bernardis, and Maurizio Ferrari Dacrema. 2020. Multi-Objective Blended Ensemble For Highly Imbalanced Sequence Aware Tweet Engagement Prediction. In *Proceedings of the Recommender Systems Challenge 2020* (Virtual Event, Brazil) *(RecSysChallenge '20)*. Association for Computing Machinery, New York, NY, USA, 29–33. https://doi.org/10.1145/3415959.3415998

[10] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. 2022. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in neural information processing systems* 35 (2022), 507–520.

[11] Andrey Gulin, Igor Kuralenok, and Dmitry Pavlov. 2011. Winning The Transfer Learning Track of Yahoo!'s Learning To Rank Challenge with YetiRank. In *Proceedings of the Yahoo! Learning to Rank Challenge, held at ICML 2010, Haifa, Israel, June 25, 2010 (JMLR Proceedings, Vol. 14)*, Olivier Chapelle, Yi Chang, and Tie-Yan Liu (Eds.). JMLR.org, 63–76. http://proceedings.mlr.press/v14/gulin11a.html

[12] Dietmar Jannach, Gabriel de Souza P. Moreira, and Even Oldridge. 2020. Why Are Deep Learning Models Not Consistently Winning Recommender Systems Competitions Yet? A Position Paper. In *Proceedings of the Recommender Systems Challenge 2020* (Virtual Event, Brazil) *(RecSysChallenge '20)*. Association for Computing Machinery, New York, NY, USA, 44–49. https://doi.org/10.1145/3415959.3416001

[13] Manu Joseph and Harsh Raj. 2024. GANDALF: Gated Adaptive Network for Deep Automated Learning of Features. arXiv:2207.08548 [cs.LG] https://arxiv.org/abs/2207.08548

[14] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 3146–3154. https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

[15] Marvin Minsky and Seymour Papert. 1969. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA.

[16] Liudmila Ostroumova Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. CatBoost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 6639–6649. https://proceedings.neurips.cc/paper/2018/hash/14491b756b3a51daac41c24863285549-Abstract.html

[17] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv* abs/1910.01108 (2019).

[18] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. *CoRR* abs/1708.05123 (2017). arXiv:1708.05123 http://arxiv.org/abs/1708.05123

[19] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. arXiv:2002.10957 [cs.CL]

[20] David H. Wolpert. 1992. Stacked generalization. *Neural Networks* 5, 2 (1992), 241–259. https://doi.org/10.1016/S0893-6080(05)80023-1

[21] Chuhan Wu, Fangzhao Wu, Yongfeng Huang, and Xing Xie. 2022. Personalized News Recommendation: Methods and Challenges. arXiv:2106.08934 [cs.IR] https://arxiv.org/abs/2106.08934

[22] In-Kwon Yeo and Richard A. Johnson. 2000. A New Family of Power Transformations to Improve Normality or Symmetry. *Biometrika* 87, 4 (2000), 954–959. http://www.jstor.org/stable/2673623