POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Exploiting LMs for Clickbait Detection in Online News Recommendation

Tesi di Laurea Magistrale in
Computer Science and Engineering - Ingegneria Informatica

Saverio Maggese, 10886750

Abstract:
As online news consumption grows, effective news recommendation is increasingly essential, balancing user interests with ethical considerations. However, news recommendation is less studied compared to movie or product recommendation, partly due to limited benchmark datasets. Many technical challenges are faced in this area, going from the rapid decay of news relevance to the use news' textual content. *Clickbait Detection*, intended as the identification of those articles in which the preview of the article differs from its content, is among such challenges. News Recommendation was the topic of the ACM Recsys Challenge 2024 organized by Ekstra-Bladet. In such context, our purpose as Politecnico di Milano's team was to develop a winning solution at predicting the user click in a given impression. This thesis will go through all the work that we conducted as a team to reach the solution that allowed us to win the academic leaderboard, detailing the data analysis and the features we obtained from the dataset, focusing on the ones that lead to a more significant performance boost. We will also dedicate attention to the models that we used and how we combined their predictions to maximize the final accuracy.
Later we will focus on 4 research questions that emerged after the Challenge completion RQ1) How do the recommendation lists change when we encode different textual parts of the articles and feed them to the recommendation algorithms?, RQ2) Does the previous results change when we only consider correct recommendation?, RQ3) Are the results influenced by the length of the encoded textual part?, and RQ4) How does the choice of the encoded part impact the overall recommendation trends? To answer these research questions we compared the recommendation lists obtained by 3 recommendation algorithms when fed with the embeddings of different parts of the articles, obtained by using the Language Models as Sentence Encoders.
Firstly, our goal is to verify that recommendations made using title encoding differ from those using content encoding, if this happens we could infer the presence of a *clickbait* phenomenon (RQ1). Secondly, we want to verify that the results of the previous point depend on the semantic difference between title and content and not on other factors such as the modest accuracy of the models (RQ2) or the length of the encoded components(RQ3). Finally, we want to identify individual candidate *clickbait* articles, recognizing them in those articles that are globally recommended significantly more times when title encoding is used compared to when content encoding is used(RQ4).
For RQ1 we compared the correlation scores of the recommendation lists, to verify if the recommendation algorithms that leverage the title encoding are less correlated with the ones that use other article's parts. At the end of the first experiment we found that our hypothesis held and therefore Language Models' strong semantic understanding can be used to detect *clickbait*. To test RQ2 we repeated the previous analysis considering only the distribution of correct recommendations, since the results were comparable to those of the first experiment we understood that the recommender accuracy does not compromise the results of the previous point. In the third experiment we verified if there was a linear or monotonic behavior between the correlation scores and the encoded parts' lengths. The experiment showed there was not, confirming that the differences found in the previous points are solely to be considered as a result of the semantic gap between the title and the rest. Finally, in the last experiment, we understood how the general recommendation trends vary, noting some exceptional behavior by the recommenders that use the encoding of the article. In other words we found that the choice of the encoded parts influences not only the ranking of the articles in the same recommendation lists, but also the global behavior of the recommender systems that tends to privilege what we consider candidate *clickbait articles*.

Advisor:
Prof. Maurizio Ferrari Dacrema

Co-advisors:
Andrea Pisani PhD

Academic year:
2023-2024

Key-words: ACM RecSys Challenge 2024,Clickbait,Embedding, Language Models,News Recommendation

1. Introduction

Many online platforms rely on Recommendation Systems(RS) to enhance user satisfaction by delivering content that aligns with their preferences. News platforms are no exception, as they leverage sophisticated recommendation algorithms to improve user experience and engagement. These systems help users discover articles that suit their interests, making it easier for them to browse through vast amounts of content.
The ACM RecSys Challenge 2024 [1], organized by Ekstra-Bladet, set out to identify the most effective approaches for predicting the likelihood of user clicks on news articles within a given impression. The term impression is used to describe the instance when a set of articles is displayed to the user during his browsing session. This prediction task presented several challenges, including handling a large-scale dataset and accurately evaluating model performance under constrained computational resources. These difficulties required creative solutions to balance model complexity with practical feasibility. As the Politecnico di Milano university's team[5][12][9][16] in the Challenge, we contributed to develop an optimized stacked ensemble model that could effectively exploit a diverse set of features. Our solution integrated predictions from multiple models, enabling us to better capture user behavior and preferences. This approach allowed us to achieve the first among academic teams and sixth overall in the competition. Our success was a result of strong collaboration, with each team member focusing on different aspects of the Challenge while contributing to the project as a whole.
My primary responsibility within the team was to explore how Language Models(LMs) could be utilized to extract valuable information from the textual content of the articles. Given the ability of LMs to understand and process natural language, I tried to leverage this capability to enhance our feature set. Throughout the Challenge, we experimented with various strategies for integrating textual information into our models. Among these, the most effective approach involved encoding the textual content of articles into embeddings — numerical representations that capture semantic information — and incorporating these embeddings as contextual features in our RS. This technique significantly boosted the accuracy of our models.
Post-competition, specific aspects of our solution were inspected, particularly concerning embedding applications in news recommendation. Our focus centered on leveraging LMs to address *clickbait detection*, a key factor in ensuring that recommendations maintain quality by delivering relevant, substantive content instead of misleading, click-driven headlines.
Clickbait refers to online content, such as headlines, images, or links, that are designed to grab attention and entice users to click on them, often through sensationalism, misleading information, or exaggerated claims. The primary goal of *clickbait* is to generate web traffic or increase engagement, rather than provide valuable or accurate content. While *clickbait* can boost the number of views or clicks, it often leaves users disappointed, as the actual content usually does not live up to the expectations set by the headline or preview.

2. EbNeRD dataset

During the ACM RecSys Challenge 2024, we used EbNeRD[29],a large-scale dataset that represents a significant contribution to the field of News Recommendation research, specifically curated to facilitate advancements and establish benchmarks. This data collection has been developed by Ekstra-Bladet, a prominent Danish news tabloid, with the goal of supporting the improvement of recommendation algorithms and systems.
EbNeRD is characterized by its comprehensive scale,it gathers data from over 2.3 million users and more than 380 million impression logs. These logs have been collected from Ekstra-Bladet during a six-week period, spanning from April 27 to June 8, 2023. The specific timeframe was intentionally selected to ensure a balanced and typical user behavior pattern, avoiding major events such as national holidays, elections, or other significant occurrences that might induce unusual or unrepresentative engagement patterns on the platform. This approach aimed at ensuring reliability for the dataset's research purposes.
In recognition of the importance of user privacy, the dataset's construction involved rigorous anonymization protocols. To achieve this, user logs were anonymized using one-time salt mapping, safeguarding user identities while retaining the behavioral insights necessary for effective recommendation system research.
Beyond user interaction data, EbNeRD includes a rich collection of news articles published by Ekstra-Bladet during the observation period. These articles are supplemented with detailed textual content features, including titles, subtitles, full bodies, and categorization of the articles. This content provides a foundation for understanding the nature of the articles and their appeal to users. In addition to these basic textual features, the dataset is further enriched with proprietary model-generated features, which include topics, named entity recognition (NER), and article embeddings.
Three distinct versions of the dataset were provided for the ACM RecSys Challenge 2024: demo, small, and large. Each of these versions is organized into self-contained training and validation sets, enabling researchers

[1] Consulted on November 4, 2024 `http://www.recsyschallenge.com/2024/`

to develop and evaluate their recommendation models in a consistent and reproducible manner. The datasets include two primary types of logs: behavior logs and history logs.[3]
The behavior logs capture records of user interactions, specifically focusing on impression data gathered over a period of seven days. An impression represents the collection of articles displayed to a user during a specific session, known as the inview list, along with a record of the articles the user actually clicked on from that list. The inview lists are designed to vary in size, containing a minimum of 5 and a maximum of 100 articles, ensuring diversity in the content shown to users during their browsing sessions.
Complementing the behavior logs are the history logs, which offer a broader view of user engagement by listing the articles a user clicked on during the 21 days preceding the recording of their behavior logs. This additional data provides insights into longer-term user preferences and interests, facilitating a more comprehensive understanding of user behavior over time.
It is worth noting that several user profiling features, such as age, gender, and postcode, contained missing values, which may require special handling when building recommendation models. Despite these gaps, the dataset allows for an exploration of user behavior patterns, especially given the presence of seasonal trends in the impressions data. For instance, the dataset reveals both weekly and daily seasonality, indicating that the volume and nature of impressions vary depending on the day of the week and the time of day. A specific pattern observed in the data is a peak in user impressions around 6 a.m., reflecting user activity patterns at that time.
In terms of article preferences, the dataset indicates a pronounced user interest in articles related to the Kendt topic, which focuses on celebrities. This trend suggests a strong inclination among users towards content in this category, making it an important consideration for model development and optimization. Figure 1 shows the most popular topics.

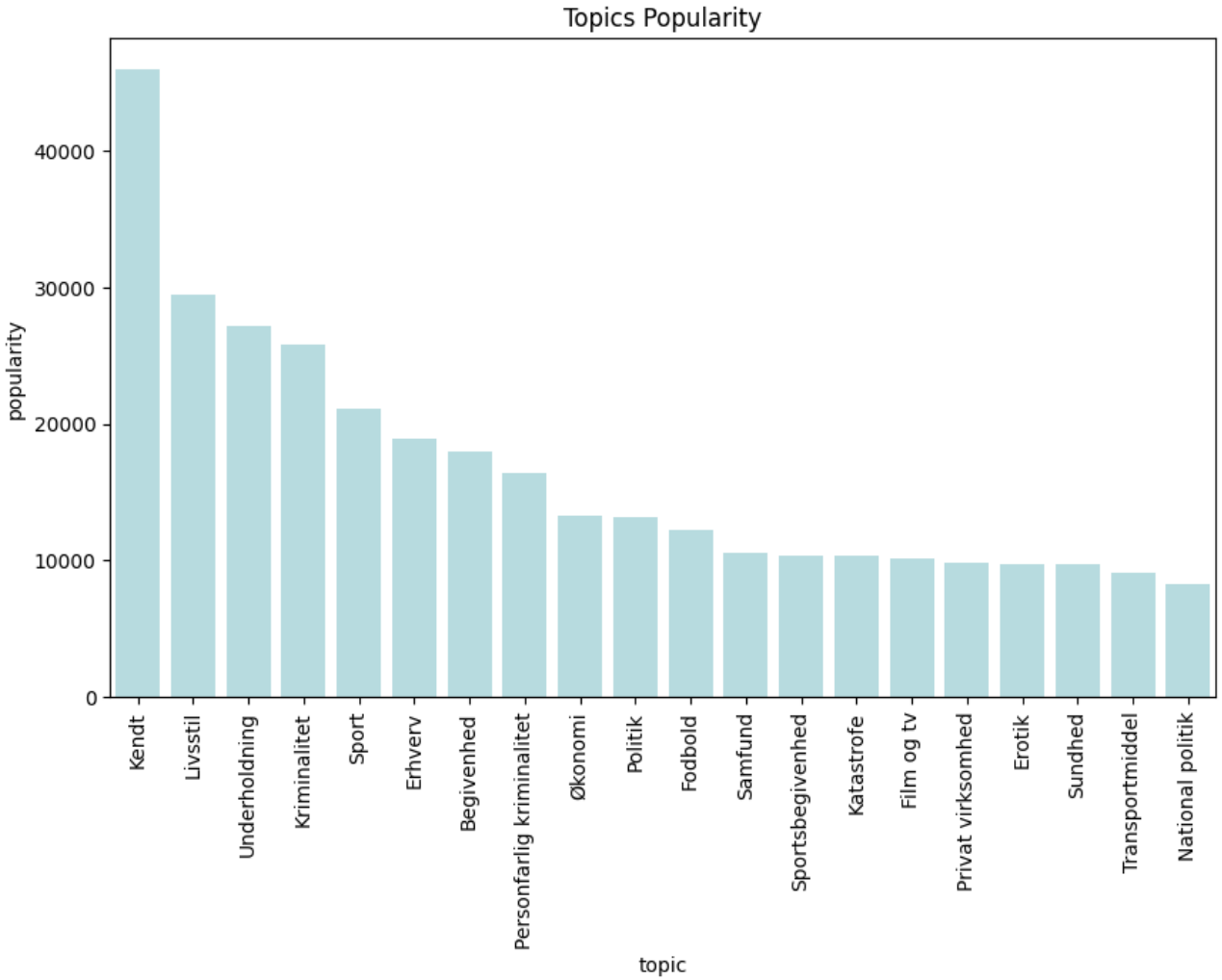


Figure 1: The number of published articles for the 20 most popular topics.

3. Problem Formulation

The primary objective of the RecSys Challenge was to develop algorithms capable of accurately estimating the likelihood of a user clicking on any given article, based on an evaluation of the alignment between the article's content and the user's inferred preferences. Participants in the Challenge were tasked with ranking articles according to these likelihood estimates. The effectiveness of their RS was measured by comparing the predicted rankings against the actual user interactions, using the Area Under the Receiver Operating Characteristic Curve (AUC) as the principal evaluation metric. AUC is a performance metric often used in binary classification tasks to evaluate a model's ability to distinguish between positive and negative classes. Specifically, it refers to the area under the Receiver Operating Characteristic (ROC) curve, which plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. The AUC value ranges from 0 to 1, where a higher AUC indicates a better model at differentiating between the classes, with an AUC of 0.5 suggesting random performance. AUC is calculated using equation 1 (Discrete Formula).

$$\text{AUC} = \sum_{i=1}^{n-1} (\text{FPR}_{i+1} - \text{FPR}_i) \cdot \frac{\text{TPR}_{i+1} + \text{TPR}_i}{2} \tag{1}$$

Among the metrics used to evaluate the models there is the Mean Reciprocal Rank (MRR)[49]. MRR is a statistic measure for evaluating any process that produces a list of possible responses to a sample of queries, ordered by probability of correctness. MRR is commonly used when the focus is on finding the highest-ranked relevant item for each query, particularly in RS and search engines. MRR is calculated using equation 2, where |Q| is the total number of queries and r_i is the rank of the first relevant result for the i-th query.

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{r_i} \tag{2}$$

4. Feature Engineering

Feature Engineering has been an important step of our work. In this section, we provide an in-depth discussion of the features we developed during the competition in addition to the features provided in the dataset. The reformatted dataset included 392 features; the most important ones are described in the following sections.

4.1. Temporal and Contextual Features

The analysis of temporal and contextual features seeks to capture the relationship between the timing, popularity, and relevance of articles or topics, and how these factors influence the likelihood of a user engaging with the content through clicks.

4.1.1 Trendiness over n Days

Each article a within the inview list is evaluated for its *Trendiness* which measures the relevance of its topics $\mathcal{T}(a)$ at the moment the impression occurs. *Trendiness* 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[3].This metric is determined by summing the occurrences of each topic in articles published during the n days prior to the impression. The objective is to assess how well an impression aligns with currently popular or widely discussed subjects. To better represent trendiness over different timeframes, we also compute and compare ratios of Trendiness scores calculated with varying values of n (e.g., comparing 1-day and 3-day trends). Additionally, using the history logs, we calculate the *Mean Topic Trendiness* for each article a, which represents the average trendiness score across all topics associated with that article. Specifically, this is formulated as:

$$MTT(a) = \frac{1}{|\mathcal{T}(a)|} \sum_{t \in \mathcal{T}(a)} \frac{1}{|I(t)|} \sum_{i \in I(t)} T(a_i). \tag{3}$$

Here, $I(t)$ denotes the set of interactions in the *history logs* involving articles containing the topic t. The term a_i refers to an article clicked by a user within a particular impression i, and $T(a_i)$ represents its *Trendiness*.

4.1.2 Endorsement

Endorsement is a feature that helps understand how much the article is popular at the moment. It calculates how many times the given article has appeared in any user impression in the hours preceding the impression timestamp.*Endorsement Article-User*, on the other hand quantifies how many times the article is appeared in the *inview list* of the same user in the last hours. These features help us understand how much the recommendation algorithm is sponsoring the article both to the users in general and to the specific user for whom we have to predict the click. These features were helpful since they tried to model the editorial choices of the platform, indeed some articles can be endorsed more than others as a result of a specific editorial decision.

4.1.3 Article Delay

To quantify how recently an article was published, we compute the time elapsed, in both days and hours, between the timestamp of the impression and the article's publication date. Additionally, we leverage the *history logs* to derive the *Mean Topic Delay*, following a method similar to that described in equation 3, but replacing the *Trendiness* metric T with the *Article Delay*. This metric aims to capture the temporal relevance of topics, indicating how long articles covering specific subjects remain appealing to users.

4.1.4 Leaking Features

For newly published articles, certain features such as *Endorsement* and *Trendiness* may provide limited insights due to a lack of sufficient historical data. Historical data is intended as for the articles the user has interacted in the *history logs*. To account for newly published articles, it is crucial to consider future user interactions and trends. We compute versions of this features that include future impressions, resulting in what we call *Leaking Features*. Furthermore, we classify as *Leaking Features* some attributes present in the dataset, such as *Total Pageviews*, which includes user interactions that occur after the considered impression.

4.2. User History Features

4.2.1 Article Topics Jaccard Similarity

A measure of topic-based similarity between articles can be obtained through the Jaccard Similarity, which compares the sets of topics for two articles. For a given user's history h and an article a in their *inview list*, we calculate the Jaccard Similarity between a and each article in h. The resulting similarities are aggregated using various statistical measures (*mean*, *max*, *standard deviation*, *min*, and others), and these aggregated values are subsequently used as predictive features. The Jaccard Similarity [23] is a measure of similarity between two sets and is defined as the size of the intersection divided by the size of the union of the compared sets. The Jaccard Similarity is calculated as in equation 4, where A and B are the sets.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{4}$$

4.2.2 Topics TF-IDF Cosine Similarity

Term Frequency–Inverse Document Frequency (TF-IDF) is a statistic used to evaluate the relevance of a word in a document relative to a collection of documents. TF-IDF is calculated according to equation 5 Another method for assessing the similarity between a user's history and an article a is through the use of TF-IDF-based cosine similarity. A TF-IDF vectorizer is trained on the topics of all articles in the dataset, enabling the conversion of an article's topic list into a vector representation. Similarly, a user's history is transformed into a vector by combining the topics from all articles they have interacted with and processing this combined text through the same vectorizer. The cosine similarity between the resulting vectors for the user's history and the candidate article is then computed and used as a feature.

$$\text{TF-IDF}(t,d,D) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d} \times \log \frac{N}{|\{d \in D : t \in d\}|} \tag{5}$$

4.2.3 Latent Dirichlet Allocation (LDA)

A third approach for quantifying the similarity between a candidate article and a user's history employs Latent Dirichlet Allocation (LDA)[6].A 5-dimensional LDA model trained on article titles is used to generate topic distributions for both user histories and candidate articles. These distributions are then compared using cosine

similarity, and the resulting similarity scores are aggregated using the method described in 4.2.1. Additionally, for each user's history, the mean LDA embedding, which provides five continuous features representing the user's topic preferences, was computed.

4.2.4 Mean User Trendiness

User interest in current events and trending topics varies rapidly. To model this aspect of user behavior, we calculate the average *Trendiness* score of the articles in each user's history. This feature captures the extent to which a user tends to engage with content that is aligned with popular or trending topics. In other words, it quantifies how much the user is generally interested in trends.

4.2.5 Mean User Delay

The Mean User Delay feature is derived by computing the average time difference between the timestamp of each user interaction and the publication date of the corresponding article, across all interactions in the user's history. It is useful to represent the user's general preference for more recent or older content.

4.2.6 User Behavior Statistics

To better model user engagement patterns, we compute summary statistics from user histories concerning features such as reading time and scroll percentage of the articles they have interacted with. Additionally, we analyze the frequency of different article categories within a user's history. Our findings suggest that the significance of these frequencies varies across categories, with certain types.

4.3. Embeddings and ICM Features

The Challenge organizers provided all participants with four artefacts containing articles' embeddings obtained with four different models (namely, the multilingual BERT[14], RoBERTa[32], a proprietary contrastive-based model and Word2Vec[33])computed by concatenating the articles' title, subtitle and body. Additionally, we used Hugging-Face encoders to build new embeddings. In particular, we used the Hugging-Face multilingual DistilBERT [40] to generate 768-dimensional title embeddings, then we concatenated titles and subtitles and used a Danish version of the MiniLM [47] to generate a 512-dimensional dense embedding. The goal of such approach was to give more relevance to the title and subtitle presuming these were more useful for predicting clicks since a user only sees the title from the user interface. Moreover, a 6-dimensional vector representing the emotions,in particular each dimension is to be thought as a score to the following emotions: sadness,joy,love,anger,fear,surprise. This embedding captures the emotions expressed in the title and subtitle and has been extracted in order to better exploit the intrinsic semantics contained in the provided text. We then used the embeddings to build an Item Content Matrix (ICM) for each embedding kind, having the articles' IDs as rows and a column for each dimension of the embedding. The ICMs were then used as inputs, together with the implicit User Rating Matrix (URM) for a pure content based (PureCB) recommendation algorithm to exploit the embedding semantics and improve over the pure collaborative filtering (CF) recommenders. The score produced by the PureCB recommender was then used as a feature.

4.4. Feature Normalizations and Aggregations

The computed features can have different ranges of values depending on many scenarios, like the hour of the day, the day of the week, or the type of user. For example, *Endorsement* with $m = 10$ can have very different values depending on the hour of the impression, e.g. it is much lower at 6 a.m. than at 7 p.m., since the 10-hour period before 7 p.m. contains many more impressions. However, it is more interesting to compare the value of a feature for a candidate article with the values of the same features in the other articles of the *inview list*. To handle such issues, some of the features were normalized in several ways over the impression id, the user id and the article id. Said normalizations are described in Sections 4.4.1, 4.4.2, and 4.4.3. The unnormalized feature values were not discarded.

4.4.1 Max Normalization

To compute Max Normalization, the value of each feature is divided by the maximum value found within the selected group of rows. This normalization ensures that the range of feature values remains comparable across different groups. For features that only take positive values, this is equivalent to applying an l_∞ normalization, effectively scaling the values relative to the group's maximum.

4.4.2 Median Subtraction

To assess how a specific feature value for an article deviates from other articles within the chosen group of rows, the median value of the group was subtracted from the article's raw feature value. The median is chosen over the mean due to its greater resilience against outliers, ensuring that the resulting values more accurately reflect typical differences within the group.

4.4.3 Ranking

Ranking involves assigning an ordinal value to each article based on the value of the feature under consideration. Articles are ranked in either descending or ascending order, depending on the nature of the feature and the relevance of its order. For instance, a descending order is used for features like *Endorsement*, where higher values are more meaningful, while an ascending order is applied to features such as *Article Delay*, where lower values might be more significant. This ranking approach provides a relative measure of where an article stands within its group for a given feature.

4.4.4 Feature Aggregations

To enrich the contextual understanding provided to our models,various aggregations, such as standard deviation, skewness, kurtosis, and entropy, of feature values were computed across their respective *impression IDs*. The purpose of the derived features is to encapsulate the distribution of feature values within the *inview list* of a given impression, offering a more detailed view of how the values are spread in the *inview list*.
Additionally, to leverage the temporal dynamics inherent in the *Endorsement* feature, this attribute was processed through several specific methods:

Temporal Sum Normalization: the *Endorsement* of each article was divided by the total *Endorsement* of all articles observed within the same m-hour time window. The resulting normalized feature represents an l_1 normalization, allowing it to be interpreted as the proportion of user u's impressions during the past m hours that included article a in the *inview list*.

Temporal Quantile Normalization: the *Endorsement* value was normalized by dividing it by the α-quantile (specifically, the 80th percentile) of the *Endorsement* distribution across articles within the same time window to prioritize the use of the 80th percentile rather than the maximum value.

Rolling Average Subtraction: To capture evolving temporal trends, the rolling average of an article a's *Endorsement* was subtracted across a predefined time window w_1 to highlight fluctuations relative to the recent trend. Additionally, we introduce a further feature by substituting the raw *Endorsement* value with a second rolling average, calculated over a longer time window $w_2 > w_1$.

4.5. Feature Selection

The feature engineering phase resulted in a 392-feature dataset. Given the data size (EbNeRD comprises over 2.7 million users and more than 600 million impression logs),it was beneficial for the team to create a lighter version by keeping only the features that significantly impacted the model's AUC in order to reduce the computing time and limit the memory usage. We performed feature selection using the null importance method [4]. As a first step, the null importance distributions were created: fitting the model over several runs with different versions of the dataset,where ,at each fitting, the target column was shuffled, showing how the model can make sense of a feature irrespective of the ground truth. Then, the model was fitted on the original target and the feature importances, obtained using Information Gain[38],were gathered,allowing to get a benchmark whose significance can be tested against the Null Importance Distribution: as a result, a new importance score was given to each feature in the dataset. Using these scores allowed to identify the 256 crucial features and the remove the remaining 136: the reduced dataset was employed for training particularly resource-intensive models and for the second Tier of the stacking model.

5. Models

In the final solution were adopted both Gradient Boosting Trees (GBTs) and Neural Networks. As mentioned in Section 3:the problem was approached from two perspectives: as a classification task, aiming to predict the probability that a user would click on a specific candidate article, and as a ranking task, focusing on estimating the relevance of each article within a given *inview list*.

5.1. Gradient Boosting Trees

GBTs are widely recognized as the state-of-the-art approach for handling tabular data due to their robust performance and relatively short training times [20][25].Two GBT implementations were used: *CatBoost* [37] and *LightGBM* [27], evaluating the models in both their classification and ranking configurations as depicted in figure 1.The classification models were optimized using the log-loss function[44] to enhance prediction accuracy, while the ranking models employed distinct ranking algorithms—YetiRank for *CatBoost*[21] and LambdaRank [7]for *LightGBM*—to improve the ordering of articles within the *inview list*.

5.2. Neural Networks

The neural networks were trained for classification, using the binary cross-entropy loss function [19] and shown in equation 6.

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)\right) \tag{6}$$

For all the Neural Networks, feature distributions were preprocessed using the Yeo-Johnson transformation [52]. The following four architectures were implemented:

- *Multi-Layer Perceptron* [34]: feedforward neural network consisting of multiple layers of neurons, with batch normalization [24] and dropout [42] layers in between.
- *GANDALF* [26]: a recent deep learning architecture that exploits a feature representation learning component, named Gated Feature Learning Unit (GFLU), based on a gating mechanism and a learnable mask for soft selection of the important features.
- *Deep&Cross* [45]: a neural network model, often employed in click-through rate (CTR) scenarios, that merges deep learning with cross layers to capture complex relationships between features. In particular, we implemented DCN-V2 [46], which proposes a more advanced feature crossing layer than its previous implementation.
- *Wide&Deep* [10]: a hybrid neural network model that integrates both a wide (linear) and a deep (non-linear) component.

5.3. Ensemble Approach

To optimize the predictive capabilities of the recommendation system, a two-tier stacking ensemble approach [50] was implemented, which integrates the diverse strengths of multiple models. This method allows to combine the predictive outputs of various models in a structured manner, leveraging the unique characteristics of each model to achieve a more accurate outcome. The models whose predictions we want to combine are called Tier 1, on the other hand the Tier 2 model is the model that combines the predictions of the Tier 1 models to generate the final result.

The training process for the ensemble is divided into two key stages. In the first stage, all Tier 1 models,both GBTs (Section 5.1) and Neural Networks (Section 5.2), are individually trained using the training dataset. Once the training of Tier 1 models is complete, they are used in inference mode to generate predictions on the validation dataset. The resulted predictions are not used directly; instead, they undergo a normalization process based on *impression*, *article*, and *user ID*; said normalization ensures that the predictions are consistent and comparable across different contexts, reducing potential biases that may arise from varying scales or distributions.

In addition to the normalized predictions, there are important metrics extracted from these Tier 1 outputs, such as the rank position of each prediction within an impression, generating derived features that combined with the original selected features outlined in Section 4.5,construct an enriched dataset. This dataset serves as the input for the training of the Tier 2 model.

In the second stage, we focused on selecting the most effective model for the Tier 2 training. We experimented with both a LightGBM classifier and a CatBoost classifier for this purpose. After thorough testing and analysis of their respective performances, the CatBoost classifier emerged as the preferred choice, providing the most accurate results in terms of click prediction and article ranking. A comparison between the two is reported in Table 1.

Once the CatBoost model was selected for Tier 2, we fine-tuned its hyperparameters to maximize its predictive accuracy. Then, the Tier 1 models were retrained using an expanded dataset, which combined both the training and validation sets. The retraining step aimed to ensure that the Tier 1 models could benefit from a larger pool of data, thereby improving their generalization capabilities. The predictions generated from the retrained

models were processed in the same manner as described earlier — normalized and augmented with rank-based metrics — before being used to construct the final input dataset for Tier 2.
This multi-tiered ensemble strategy allowed us to harness the strengths of different models, creating a unified approach that is more accurate and reliable than any individual model,as shown by the results in Table1.
We removed all the features eliminated by the Null Importances feature selection and integrated new features derived from the predictions of the Tier 1 models. To produce the predictions on the test set, we first retrained Tier 1 models on the full dataset, concatenating train and validation. Then, we used them to produce the predictions for the test set.

5.4. Hyperparameter Tuning

To optimize the performance of our models, we conducted hyperparameter tuning using the small dataset due to constraints in computational resources.
The primary metric we aimed to optimize was AUC (Section 3), as it effectively measures the ranking capability of the models and their ability to distinguish between clicked and non-clicked articles [31]. For the tuning process, we used Bayesian Optimization [35], a method known for its efficiency in exploring hyperparameter spaces, especially in scenarios with limited resources[15]. We implemented this using the Optuna framework, which is well-regarded for its flexibility and capability to manage complex optimization tasks[2].
The hyperparameter tuning process for the Tier 1 models followed the provided dataset splits, which allowed for a systematic evaluation on the designated validation set. During the data preprocessing stage, we also optimized the hyperparameters for the ItemKNN models, as outlined in Section 4.3, treating them similarly to other Tier 1 models to ensure consistency in our approach.
For Tier 2 models,w e created an additional split within the training set's behavior logs to facilitate the tuning process. Specifically, we divided the training data into two segments: the first 3.5 days were allocated for training the previously optimized Tier 1 models, while the subsequent days were used for generating predictions that served as input data for tuning the Tier 2 models. This split enabled the Tier 2 models to be trained on predictions derived from optimized Tier 1 models, thus allowing for a more accurate refinement of the Tier 2 models' hyperparameters based on the intermediate results.
This structured approach to hyperparameter tuning, as outlined by the accuracy the model was able to achieve, allowed us to maximize the predictive power of both Tier 1 and Tier 2 models, contributing to the overall effectiveness of our ensemble approach.

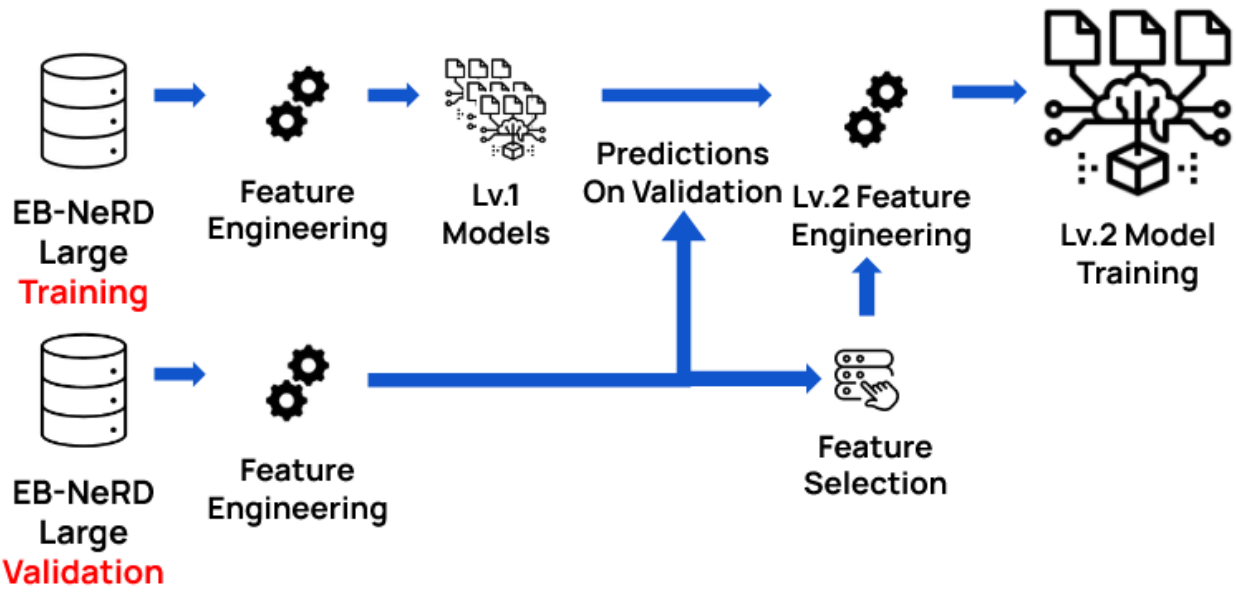


Figure 2: The whole training process: we trained Tier 1 Models on the training split and we ran inference on the validation set. These predictions are then combined with selected features extracted from the validation set. Finally, the level 2 model is trained on the predictions and features derived from the validation set

6. Results

Table 1 presents the AUC and MRR results attained by all our models when evaluated on the official test set. Among the various models assessed, Catboost emerged as the best-performing Tier 1 model. Overall,GBT models demonstrated consistently better accuracy compared to neural network architectures. However, neural network models played a significant role in the composition of our final ensemble model. Their inclusion was important as they provided diverse recommendations that complemented the outputs of the GBT models, enhancing the overall predictive capability.
The integration of all the models into our final ensemble resulted in a notable improvement, yielding an increase of 0.71% in the AUC score respect to the best Tier 1 model, namely Catboost Ranker. Ultimately, the highest AUC score achieved in our experiments was 0.8513, reflecting the robustness and accuracy of our ensemble methodology.

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

We conducted an additional ablation study focused specifically on the Tier 1 Catboost Classifier model to evaluate the influence of various components incorporated within our project. The methodology involved establishing a baseline model using the small dataset, that is a model trained without the normalized features. Subsequently, we integrated the feature normalizations discussed in Section 4.4.
The findings are displayed in table 2, which indicates that the incorporation of feature normalizations applied to our initial raw features on the official test set resulted in a notable increase in the AUC metric, specifically an enhancement of 0.0809, which corresponds to 11.02%. Moreover, the transition to training on the full dataset yielded an additional improvement in AUC of 0.0209, which corresponds to an additional 2.56%.

Table 2: Impact of features normalization on the test set.

Model	AUC	MRR
Catboost Classifier (Baseline)	0.7344	0.5088
+ Normalizations & Statistics	0.8153	0.6076
+ Full Dataset	0.8362	0.6401

In Section 4.1.4 we extensively talked about *Temporal Leaks*, here we wanted to evaluate the impact of *Temporal Leaks* features on the performance of our models, we constructed an alternative version of the dataframe that excluded all features associated with temporal leakage. Then, the modified dataframe was used to train simplified models,whose performance was compared to that of identical models trained on the complete dataframe, which included the temporal leaking features.
Given the significant computational resources required to conduct experiments on the complete dataset, we opted to use the small dataset for our analysis[3]. This choice was made because the small dataset has consistently demonstrated results that align closely with those of both the complete and larger datasets.
Table 3 shows the results of the analysis on *Temporal Leaks*. From our findings, we can conclude that the inclusion of leaking features contributed to an increase in the AUC metric by approximately 0.02 to 0.03 points, contingent upon the specific model employed. It is important to note that the AUC score obtained without the leaks should be considered as a baseline value. This is due to the fact that the models were trained using the optimal hyperparameters identified for the full set of features, which means that these hyperparameters may

not necessarily be optimal once the leaking features have been removed. In other words if the difference in performance can partially be due to models' hyperparameters, which were tuned on the dataset that contained all the features. A fairer comparison would have involved an hyperparameter tuning on the dataset obtained by removing *Temporal Leaks*, by doing so the comparison would have involved both models with an optimized configuration.

Table 3: 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

6.1. Feature Importance

Figure 3 illustrates the importance of the various features employed in the Tier 1 Catboost Classifier model. Among all features under analysis, *Endorsement* and *Endorsement Article-User* emerged as the most significant predictors of click probability. Notably, the normalized versions of *Endorsement* exhibited greater significance than the raw values, which validates the reasoning outlined in Section 4.4, in particular normalization is effective since it captures the contextual component of the problem. By normalizing the features and comparing them with the ones in the same inview

In addition, *Article Delay* was identified as a critical feature, ranking highly in various forms, indicating its relevance in modeling user behavior and article relevance over time. Furthermore, features that are categorized as *Leaking Features*, such as ratio between *Total Pageviews* and *Total Inviews*, ranked as the seventh most important predictor, emphasizing the impact that these features have on the model's predictions despite their potential drawbacks in a live environment.

Content-based features, including the *Article Topics Jaccard Similarity* and features derived from the ICM, were also notable in terms of importance, securing positions 13 and 34, respectively. This suggests that also the content attributes of articles are relevant in driving click predictions.

Overall, our findings indicate that the *Median Subtraction* and *Max Normalization* techniques were the most effective normalization methods employed in our feature set, in conjunction with standard deviation aggregations. This further reinforces the notion that well-designed preprocessing and normalization strategies play a vital role in enhancing model performance as also testified by[8][48][41]

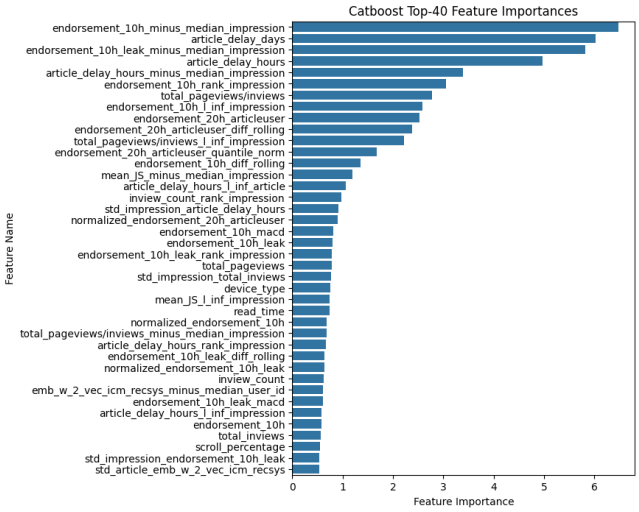


Figure 3: Top 40 Features Importance in Tier 1 Catboost

7. Challenge Conclusions

The ACM RecSys Challenge 2024 focused on predicting user click likelihood for candidate articles within specific impressions in a news recommendation setting. Our solution integrated information from article features, user history, and behavior patterns. We utilized multiple model, such as GBTs and Neural Networks, achieving enhanced prediction accuracy. Normalizing features across impressions was helpful in enhance performance, as it helped leverage contextual information for click prediction. Through a stacking ensemble, we improved individual model performance, achieving 1st place among academic teams and 6th overall in the competition.

8. Post Challenge Conclusion

During the months we spent working on the challenge as a team, there was no strict division of roles; instead, each team member contributed to various aspects of the project. However, during the competition, my primary focus was on leveraging embeddings and LMs, which played a significant role in improving the accuracy and overall performance of our final solution,as mentioned in 4.3
After the challenge, we empirically explored the following questions, stemming from the use of LM embeddings of the news articles we had worked on:

RQ1 How do the recommendation lists change when we encode different textual parts of the articles and feed them to the recommendation algorithms ?
RQ2 Does the previous results change when we only consider correct recommendation?
RQ3 Are the results influenced by the length of the encoded textual component?
RQ4 How does the choice of the encoded part impact the overall recommendation trends?

9. Experimental Setup

9.1. Datasets

We conducted a series of experiments using two datasets: the Microsoft News Dataset (MIND) [51] and the small partition of the EbNeRD dataset, used during the Challenge. MIND is a large-scale dataset specifically designed for news recommendation research, compiled from anonymized behavior logs on the Microsoft News website. Its primary goal is to serve as a benchmark for news RS, advancing research in this domain. The dataset consists of approximately 160,000 English news articles and over 15 million impression logs generated by 1 million users. Each news article is rich in textual content, providing fields such as the title, abstract, category, and entities. The impression logs include records of user interactions, including click events, non-clicked impressions, and the historical news click behaviors prior to each impression.
We selected MIND for a number of key reasons. Firstly, its structure closely mirrors that of the dataset used in the Challenge, making it an ideal candidate for comparative analysis. Secondly, MIND is one of the few publicly available datasets that offers both user interaction logs and the textual attributes of articles, making it uniquely suited to our study of embeddings and click predictions. For instance the Globo.com dataset, introduced in [11][36] provides the behaviors logs and article embeddings already computed, but not the textual components of the articles. Unfortunately we could not get access to the Plista Dataset[28].The Adressa Dataset[22] came in a different structure, therefore his integration would have been more costly with respect to MIND.
While we have already provided an overview of the EbNeRD dataset, it is useful to highlight a few details to better understand the experimental setup. We used the small partition of the dataset to mitigate the computational demands associated with the larger version.
Though MIND and EbNeRD share structural similarities, some important differences must be acknowledged. The most notable discrepancy lies in the textual fields available: whereas the EbNeRD dataset provides title, subtitle, and body, MIND only includes the title, abstract, title entities, and abstract entities. This difference is significant given that our approach involves encoding text fields and comparing their influence on recommendation effectiveness across both datasets.
Therefore, we mapped the abstract field in MIND to the body field in EbNeRD, reasoning that the abstract functions as a condensed version of the article's content and would serve as an appropriate surrogate. In addition, we concatenated the title entities and abstract entities in MIND to approximate the subtitle field in EbNeRD. Our rationale was that the entities encapsulate the most pertinent elements of both the title and the abstract, so concatenating them would generate a representation analogous to a subtitle—sitting between the title and body in terms of semantic richness.
It is important to note that more sophisticated techniques could have been employed to bridge this gap, such as using a Large Language Model to infer a subtitle for the MIND articles. However, we opted against this

approach due to the significant computational overhead it would introduce, as well as the risk of contaminating our results. Instead, we chose to maintain a simpler, more controlled approach to ensure that the results remained as reproducible and interpretable as possible.

9.2. Language Models

In line with what was done during the Challenge, we kept using LMs to encode the textual components of the articles into dense latent vectors, which we then used as input for the RSs. By transforming text into embeddings, we wanted to provide the RS with a meaningful representation that can help capture the similarities and differences between articles based on their content.
Regarding the choice of LMs, we opted for **BERT-base-multilingual-cased** model[14] and **DistilBERT-base-multilingual-cased**[40], to generate 768-dimensional dense vectors. The rationale behind such selection is twofold. First, having already employed these models during the Challenge, we could reuse some of the embeddings we had previously computed, thereby saving time and computational resources. Additionally, despite being slightly older, such models have been extensively tested and validated, providing a solid baseline for our experiments.[1]
Although more recent and advanced Language Models could offer superior text encoding capabilities, they come with certain trade-offs. Newer models often produce embeddings with higher latent dimensions, which can demand more computational resources and time for processing, especially in large-scale datasets. Moreover, many of these models have limitations on the number of tokens they can encode at once, which may have introduced additional complexity to our workflow. By keeping consistent with the models we had already tested, we were able to conduct our experiments more efficiently and without additional costs.
The use of newer models is probably worth exploring as a potential next step, as detailed in Section 15.

9.3. Recommendation Algorithms

We employed several RSs to evaluate in the context of our experiments. First, we made use of a pure content-based algorithm using ItemKNN (PureCB-KNN Recommender) [13], which leverages the similarity between articles based on their textual features. In this approach, recommendations are made by identifying articles that are most similar to those the user has interacted with in the past, relying on K-Nearest Neighbors [17] to rank the articles based on their content similarity. Next, we experimented with a Factorization Machines-based (FM) [39] approach using LightFM [30], a hybrid model that combines both content-based and collaborative filtering techniques. LightFM implements FMs to model both user-item interactions and the attributes of the articles to generate recommendations. Finally, we employed a hybrid algorithm that combines the strengths of ItemKNN with Collaborative Filtering, from now referreed as Hybrid-KNN Recommender. All three algorithms use two key inputs: the ICM and the Implicit URM. The ICM is constructed by stacking the embeddings of the articles vertically, where each dimension of the resulting vector corresponds to a floating-point value representing an attribute in the dense space of the embeddings. The URM is derived from user interaction histories, assigning a value of 1 when a user has interacted with an article (i.e., clicked on it) and 0 otherwise. It is important to acknowledge that we used traditional algorithms, that, while effective in certain contexts, are not state of the art when measured in terms of AUC, nor did they serve as highly competitive standalone models during the Challenge. The primary limitation of these algorithms is their inability to capture complex information, such as temporal dynamics or sequential user behaviors, which are often essential for improving recommendation accuracy in a session-based setting. Although more sophisticated architectures could have yielded better predictive performance, accuracy was not the primary focus of this second phase of experimentation. The objective here was to explore the role of article embeddings in recommendation tasks. More complex models typically rely on a broader set of features, which could marginalize the role of embeddings in predicting scores. During the Challenge, we observed that features like the *Endorsement* and *Delay* were highly influential in the final model predictions. By adopting simpler algorithms, we deliberately reduced the overall accuracy of the models but, in doing so, allowed the semantic power of the embeddings to play a more prominent role in the predictions, making the experiments more focused on evaluating the embeddings themselves.

10. Comparative Analysis on the Recommendation Lists Based on Embedded Article Component

In this Section, our primary focus is to assess the recommendation lists of various RS when they are provided with embeddings derived from different components of the text.
Specifically, for the Ekstra-Bladet dataset, we generated embeddings for several combinations of textual ele-

ments:

- Title;
- Subtitle;
- Body;
- Concatenation of Title and Subtitle;
- Concatenation of Subtitle and Body; and
- Concatenation of Title,Subtitle and Body.

Similarly, for MIND, we produced embeddings for:

- Title;
- Abstract;
- Concatenation of Title's Entities and Abstract's Entities (Entities);
- Concatenation of Title and Entities;
- Concatenation of Entities and Abstract; and
- Concatenation of Title,Entities and Abstract.

Based on the embeddings computed from different textual components, we created distinct ICMs, each incorporating diverse information depending on which article component was embedded. Then the ICMs were provided as input to the RSs outlined previously, generating a range of model variants. After creating the Implicit URM, we proceeded to fit the RS on the training splits of the datasets.
To ensure a set of consistent experiments we always used the same standard hyperparameters that resulted from a preliminary tuning phase. In Tables4,5 and 6 are reported the hyperparameters used for each RS.

Table 4: Hyperparameters for PureCB-KNN Recommender

Hyperparameter	**Value**
similarity	Euclidean
topK 2	1457
shrink 3	329
normalize	False

Table 5: Hyperparameters for FM Recommender

Hyperparameter	**Value**
epochs	3
loss 2	BPR
sgd mode 3	Adagrad
learning rate	0.05
number of components	10

Table 6: Hyperparameters for Hybrid-KNN Recommender

Hyperparameter	**Value**
similarity	Cosine
topK 2	50
shrink 3	100
normalize	True
ICM Bias	0

Subsequent to model fitting, we employed the trained models to perform inference over the validation sets of the two datasets. This process yielded click probability scores for each article presented in the *inview list*. Given that the number of articles in the *inview* can vary depending on the user interface, we filtered the results to retain only the top ten articles with the highest click probability scores for every impression. At this stage of our analysis, the precise click probability scores themselves became less significant, as our focus shifted away from accuracy metrics. Instead, we concentrated solely on the composition of the recommendation lists, examining the ranking of the items within them
Our primary interest lied in exploring how the recommendation lists evolved when we considered different textual components of the articles and various embedding models. In particular, we aimed to assess the extent

of change in the recommendation lists through specific comparison measures. To achieve this, we employed two key metrics: the Jaccard Similarity (JS) and the Kendall-Tau Rank Correlation (KT) [43]. KT is calculated as in the following equation 7, where n is the total number of pairs.

$$\tau = \frac{\text{Number of Concordant Pairs} - \text{Number of Discordant Pairs}}{\frac{1}{2}n(n-1)} \tag{7}$$

For each impression in the validation set, we computed the correlation metrics between each recommendation list obtained by the different models. Then, we averaged the results and visualized the relationships among the recommendation lists using a heatmap, that serves as a valuable tool to display the correlations across the recommendation lists.
Results for PureCB-KNN Recommender are reported in Section 10.0.1; Section 10.1 shows those relative to FM Recommender, while Section 10.2 shows those achieved by Hybrid-KNN Recommender.

10.0.1 PureCB-KNN Recommender

In this Section, we report the results obtained in the experiments involving PureCB-KNN Recommender.

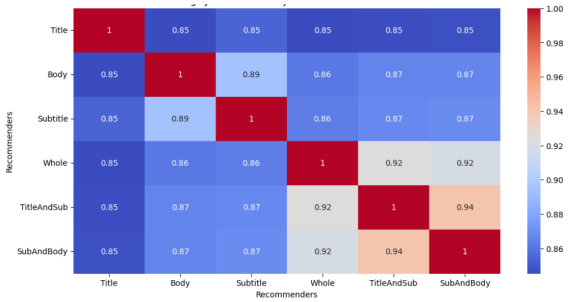


Figure 4: Comparison of JS using PureCB-KNN Recommender models on the EbNeRD dataset.

In figure 4 the comparison between the average JS obtained by the different models is shown. Different model variants are named after the embedded text part they used as input. We compared the recommendation lists obtained by the PureCB-KNN Recommender algorithm using the DistilBERT embeddings (4). Several observations can be clearly drawn from figure 4.
There is a notable difference between the recommendation lists obtained using different parts of the text as information to feed the recommender, indicating that the text source plays a significant role in computing predictions, at least in the case of the PureCB-KNN Recommender. Since the JS values are approximately the same when comparing results from the two Sentence Transformers, we report only the results referring to DistilBERT embeddings. The lowest correlation values, in terms of JS, occur when comparing the 'Title' recommender with others, meaning that recommendations computed using the title differ significantly from those based on other parts of the article. This discrepancy suggests a notable semantic difference between the title and the actual article content, a phenomenon that could be related to *clickbait* (Section 1). Figure 4 indicates that LMs might be used to detect clickbait titles and articles by analyzing recommendations generated from different textual sections of the article.
The JS values range from 0.85 to 0.95. Importantly, high JS values do not necessarily imply that the recommendation lists are identical; in particular, there may be significant differences in item ranking that JS alone does not capture. For instance, an average JS of 0.95 and KT of 0.22 would suggest that while the recommendation lists often contain the same articles, their rankings within these lists vary greatly. The KT analysis in Figure 5, which we discuss in the following paragraph, offers further insights into the relative ranking of items within the recommendation lists.
In Figure5 we report the heatmaps regarding the KT correlation score,that will give us an insight of the different rankings in the recommendation lists.

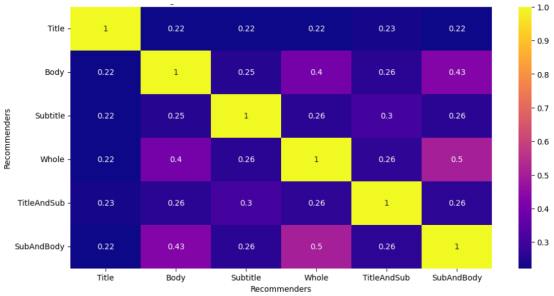


Figure 5: KT Comparison of PureCB-KNN Recommender model with BERT Embeddings on the EbNeRD dataset.

There are several observations to be made, also keeping as reference figure 4:

1. Similarly to figure 4 the results with the 2 LMs are identical, therefore we report only the results using BERT embeddings.
2. The KT scores are significantly lower than the JS scores, which highlights the relevance of the ICMs (and therefore the article parts) in computing the recommendations. In fact, the choice of the article's component not only affect the items that are in the recommendation list but, more importantly, the ranking among the same items in the recommendation list.
3. As it happened for JS, the Title model provides the recommendations that most differ from the others, emphasizing how the titles can have different semantics with respect to the other parts of the article and reinforcing the idea that some articles could represent *Clickbait*.

The MIND analysis is reported in figure 6

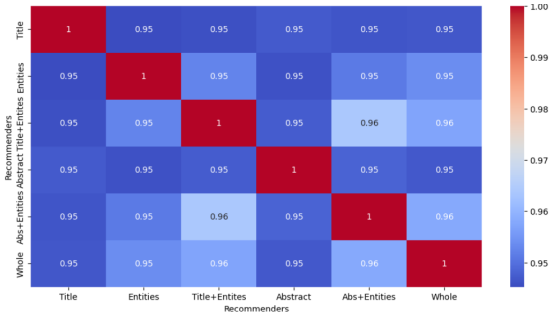


Figure 6: JS Comparison of PureCB-KNN Recommender model with BERT embeddings on the MIND dataset.

Considering the results depicted in figure 6, we draw the following considerations from what is shown in Figure 6 4. As in figure 4 we note that there is no difference in the scores obtained using BERT or DistilBERT, therefore only the BERT results are shown. The Title is still the least correlated with respect to other parts of the text; however, in this case, we note the same effect for the abstract. Abstract and title seem to be less related with each other and to the other textual components, therefore the semantics of the title and the abstract lead to different recommendations . Since the abstract can be considered as the summary of the articles' content, the results can eb considered coherent to those observed for the EbNeRD dataset. In general, the range of values reported in the heatmap changes with respect to the results in EbNeRD. In EbNeRD, the JS values ranged between 0.85 to 0.95; here, the values are notably higher and range between 0.95 and 0.96, meaning there is overall less difference among the recommendation lists. Given the previous point, it appears that even a small difference, such as 0.01, in the recommendation lists' JS values might hold practical significance due to the large number of impressions being analysed. However, to make a definitive statement regarding statistical significance, it would be essential to conduct a formal statistical test. Moreover, as we mentioned for EbNeRD, JS does not take into account the items' ranking.

We replicated the experiments also regarding the KT correlation score for the MIND dataset. We report the results in figure7.

Figure 7: KT Comparison of PureCB-KNN Recommender model with BERT embeddings on the MIND dataset.

The results depicted in figure 7 are different from the ones in figure 5. We immediately note that the values in 7 are higher from the ones in 5, going from values around 0.2 to values close to 0.9. Moreover, most of the values are very similar. We still can note that, in general, the values concerning the Title are the smallest. However article parts did not influence the rankings in the recommendation lists as much as in EbNeRD. Possibly the 2 datasets contain different types of articles, this results led us to believe that in MIND the articles' content is more coherent to the title with respect of what occurs in EbNeRD. Moreover MIND collects articles from different sources, whereas EbNeRD only contains articles from the Ekstra-Bladet platform, the heterogeneity given by different data source might also have influenced the tests' results.

10.1. FM Recommender

In this Section, we report the results obtained in the experiments on FM Recommender, coherently with what was reported forPureCB-KNN Recommender in the previous Section. In the following figure, we show the heatmap that compares, in terms of JS, the recommendation lists obtained fitting the algorithms with embeddings of different parts of the articles. Firstly, we focus on the EBNeRD dataset.

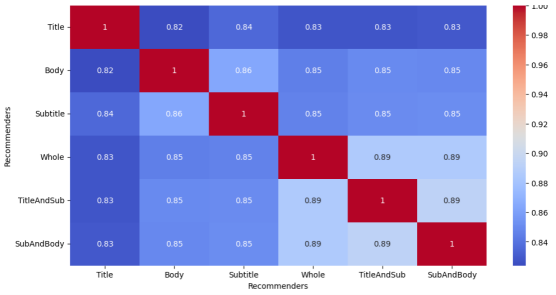
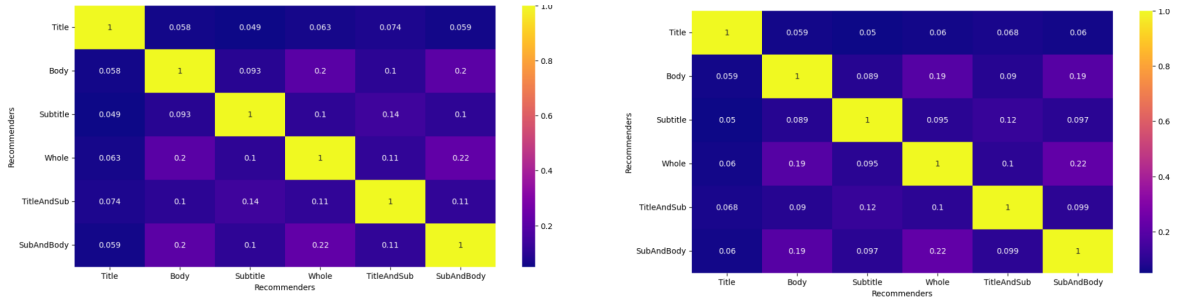


Figure 8: JS Comparison of FM Recommender model with BERT embeddings on the EbNERD dataset.

In reference to figure 8, we draw the following considerations, also considering the similarities with figure 4:

1. There still is no difference between the average JS when comparing the embeddings obtained with BERT and those obtained with DistilBERT, we report only the results with BERT;
2. As we saw in figure 4, the Title is still less related with other parts of the text; this is interesting since we could have anticipated that adding a collaborative component to the recommender would have lessened the differences. Nevertheless, the algorithm is able to capture the semantic gap between the title and the rest.

We repeat the analysis conducted in Figure 5with respect to FM Recommender.



(a) KT of FM Recommender with Distilbert embeddings

(b) KT of FM Recommender with Bert embeddings

Figure 9: KT Comparison of FM Recommender models on the EbNERD dataset.

Figure 9 shows some interesting similarities with respect to figure 5. For the first time, there is a measurable difference in the scores obtained with BERT and DistilBERT. The overall scores obtained with BERT are lower. Evidence suggests that BERT embeddings are slightly better in capturing the semantic gap between the Title and the rest. KT correlation scores are very lower than the ones depicted in figure 5. There is a noticeable difference in the semantics of the articles parts and therefore in the ranking proposed by the different algorithms, despite FM Recommender being a hybrid recommender and taking into account collaborative information. Once again, the lowest value in the grid are the ones accounting for the correlation with the recommenders that use the ICM with Title embeddings.
Results for the MIND dataset are reported in Figure 10.

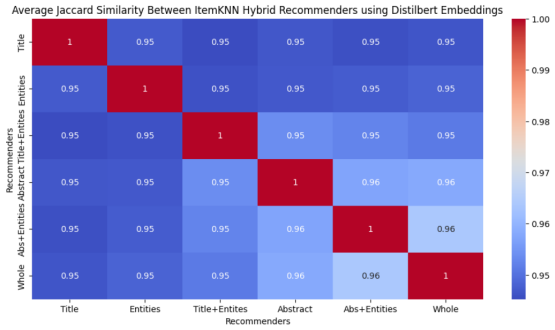
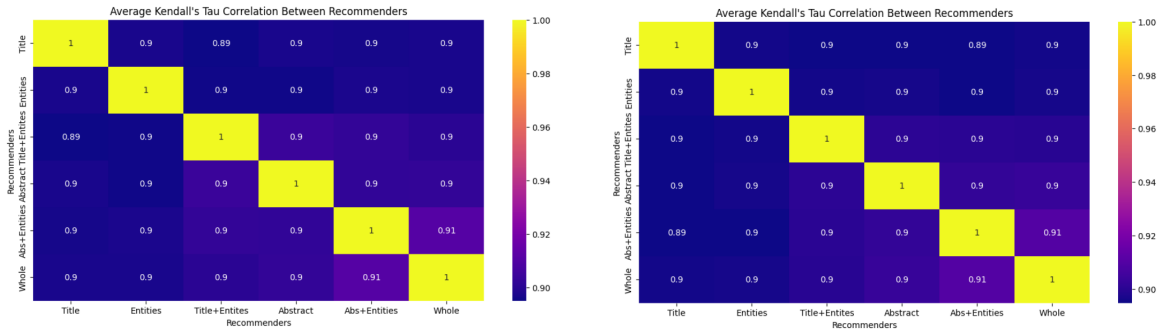


Figure 10: JS Comparison of FM Recommender model with BERT embeddings on the MIND dataset.

Figure 10 confirms our findings for PureCB-KNN Recommender in the previous Section. There are several similarities between figure10 and figure 6:

1. The values range from 0.95 to 0.96 meaning that the correlation is consistent across the board.
2. Still, we could state that the Title is the least correlated, even if by a small percentage.

KT measures for FM Recommender are plotted in Figure 9.



(a) KT of FM Recommender with DistilBert embeddings on MIND dataset

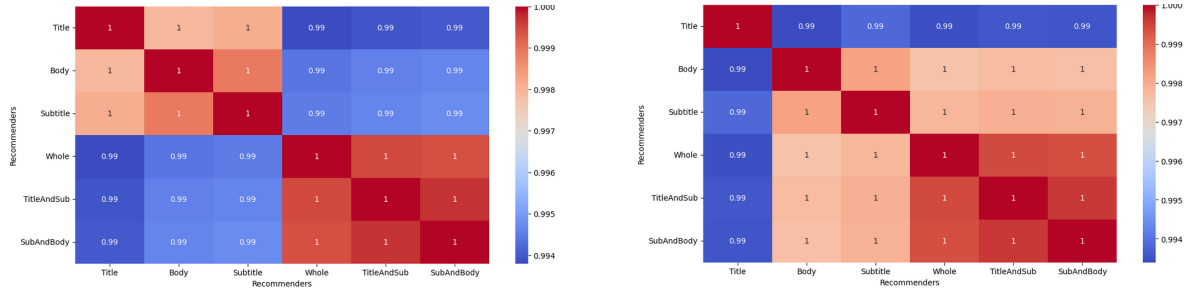
(b) KT of FM Recommender with BERT embeddings on MIND dataset.

Figure 11: KT Comparison of FM Recommender models on the EbNERD dataset.

As we appreciated for the JS in Figure 6, the values are higher with respect to those obtained for the EbNeRD dataset and more importantly are the same across the the heatmap. On the other hand, there are noticeable differences with Figure 9, where the values for the ranking metric were lower.

10.2. Hybrid-KNN Recommender

In this Section , we will report the results of the experiments conducted on Hybrid-KNN Recommender, a hybrid RS that is built on top of PureCB-KNN Recommender. More specifically, the collaborative information is added via the URM. The ICM and the URM are combined by transposing the URM and stacking it horizontally to the ICM, to form a hybrid input for the model. We will display the results keeping the same structure of the previous Sections: first focusing on EbNeRD, and then comparing the results to the ones obtained with MIND.

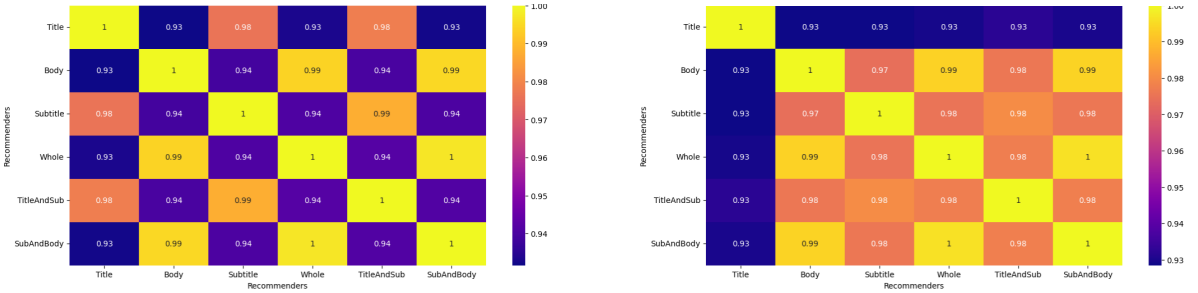


(a) JS of Hybrid-KNN Recommender with DistilBERT embeddings on EbNeRD.

(b) JS of Hybrid-KNN Recommender with BERT embeddings on EbNeRD.

Figure 12: JS Comparison of Hybrid-KNN Recommender models on the EbNERD dataset.

The results displayed in Figure 12 deviate from those observed in the Sections discussing other models, particularly, from those of FM Recommender examined in Section 10.1.
Using both sentence transformers, the data suggests that recommendation lists produced by this model remain unchanged, regardless of the specific encoded text part used in the input, indicating a dominant influence of the collaborative filtering components over the content-based features in these recommendations. As shown in Figure 12b, the correlations across most encoded article parts are nearly identical, with only the Title-encoded parts showing any deviation from a JS score of 1. This implies that the inclusion of Titles contributes to some, although limited, differentiation.
For DistilBERT embeddings, a further pattern emerges: the models using ICMs derived from concatenated article parts exhibit slightly lower correlation with respect to those based on single article parts. However, the differences are marginal, effectively rendering the impact negligible.
Data suggest a strong influence of the CF component in this hybrid model setup, which appears to limit the role of content-based features in shaping recommendation lists, despite their integration.
KT analysis is displayed In Figure 19.



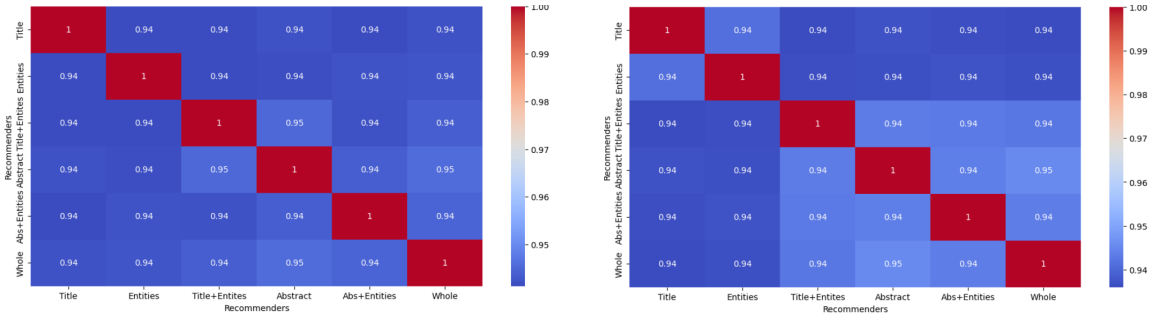
(a) KT of Hybrid-KNN Recommender with Distilbert embeddings on EbNeRD dataset.

(b) KT of Hybrid-KNN Recommender with BERT embeddings on EbNeRD dataset.

Figure 13: KT Comparison of Hybrid-KNN Recommender models on the EbNERD dataset.

As clearly visible in Figure 13 we have different results with respect to previous models. First, the results obtained using BERT embeddings are clearly different from the ones obtained using DistilBERT embeddings, as the correlation scores obtained using DistilBERT, confirming the impression we had when checking JS scores

for the same RS, as shown in Figure 12. On the other hand, the BERT embeddings replicate the usual pattern: once again the Title is the least correlated. In this Section we started to appreciate a difference in the embeddings semantics that in the beginning seemed to yield to the same results in all circumstances. The differences in the results are probably due to the size of the two models: while BERT contains around 110 million parameters, DistilBERT has only 66 million parameters. Of course, the more the parameters, the more the expressive power, which could explain why BERT is more capable to capture the semantics of the articles' parts. Experiments with the MIND dataset are reported in this Section.

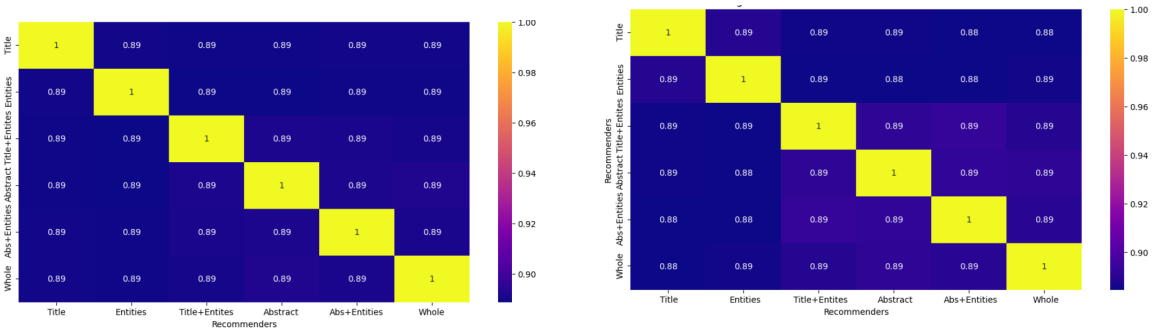


(a) JS of Hybrid-KNN Recommender with DistilBERT embeddings on MIND dataset.

(b) JS of Hybrid-KNN Recommender with BERT embeddings on MIND dataset.

Figure 14: JS Comparison of Hybrid-KNN Recommender models on the MIND dataset.

Figure 14 reports the same results we always saw when studying the MIND dataset in Figure 6 and Figure 10, the JS scores are constant for all pairs of recommenders. It could be appropriate to inspect a bit further the MIND dataset to check why this difference exists. First we inspect the results on the ranking measure.



(a) KT of Hybrid-KNN Recommender with DistilBERT embeddings on MIND dataset.

(b) KT of Hybrid-KNN Recommender with BERT embeddings on MIND dataset.

Figure 15: KT Comparison of Hybrid-KNN Recommender models on the MIND dataset.

Figure 15 results are those that we expected,as they mimic those of Figure 14.
From the analysis conducted in this Section, we can draw the following conclusions. In most of the experiments, the results obtained with BERT and DistilBERT embeddings are similar. However, for the FM Recommender and Hybrid-KNN Recommender, BERT better underlines the semantic gap between the Title and other text components. The results obtained with all the recommendation algorithms are quite coherent. The Hybrid-KNN Recommender produced slightly different results that highlighted a difference between the sentence transformers. The results obtained during the experiments, mostly the ones with the PureCB-KNN Recommender and FM Recommender, highlight a semantic gap between the articles' Title and the other parts of the same, especially concerning the EBNerD dataset. The proposed method appears to be suitable to the purpose of detection of a *clickbait* phenomenon in our data.

11. The role of Accuracy in *Clickbait Detection*

In this Section, we report the experiments of an analysis similar to the ones in Section 10. However we will consider how the recommendation lists change when considering only the correct recommendations. We consider as *correct* the recommendations in which the target article is in the recommendation lists obtained with a cutoff

of 10. By not considering incorrect recommendations, we can better test if the results we got in Section 10 were influenced by the models' accuracy. We keep only the impressions for which the models we are comparing have the target article in the recommendation list. We will neglect the results obtained with Hybrid-KNN Recommender since they add no information with respect to the analysis in Section10. Indeed we conducted the same experiments also with Hybrid-KNN Recommender and the results were fundamentally identical to those in Section 10. Possibly Hybrid-KNN Recommender predominantly bases its predictions on the URM and, therefore it would not be useful to study the results it produces. In all the experiments reported in this Section we expect an overall increment of the Correlation metrics due to 2 factors: firstly, the average Correlations are computed on less impressions, therefore in the computation the denominator is smaller and the final value is generally higher; secondly, considering only the recommendations lists that both contain the target articles means to compare more similar lists in general, leading to closer results in terms of correlations coefficients.

11.1. PureCB-KNN Recommender

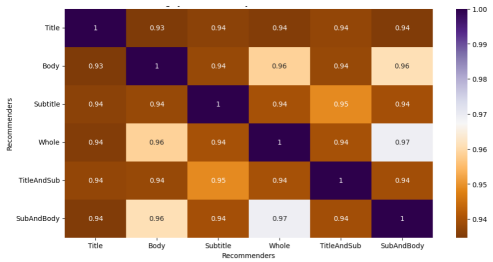


Figure 16: EbNeRD Analysis of JS for correct recommednations only in PureCB-KNN Recommenderr

In Figure 16 we depicted the JS analysis on the EbNeRD dataset considering only correct recommendations. Data shows a confirmation of what we appreciated in Figure 4. We have the same results in terms of comparison but not in terms of overall correlation scores. In figure 16, there are overall higher levels of similarity: this could lead to think that what we said in the previous Section is no more true when considering only correct recommendations, however this overall increment was to be expected (as mentioned in the previous paragraph). We proceed to inspect the results that consider KT correlation.

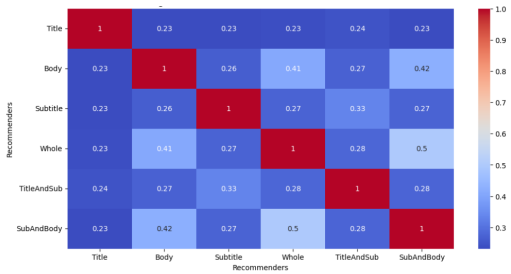
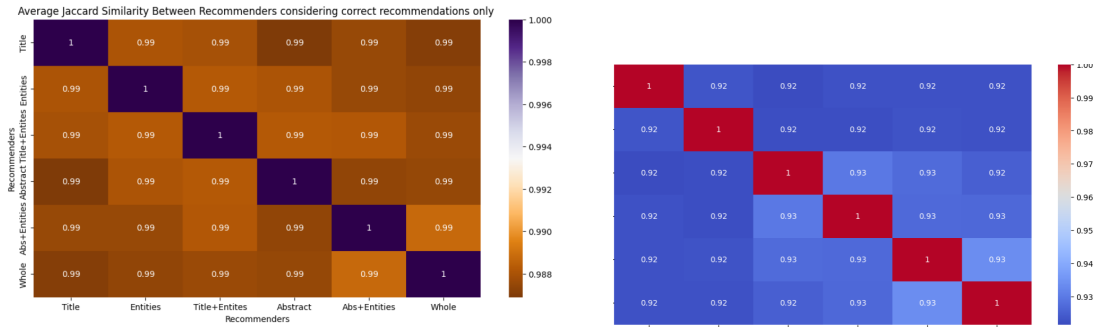


Figure 17: EbNeRD Analysis of KT for correct recommednations only

Keeping in mind the considerations that followed Figure 16, in Figure 17 we have a confirmation of the results relative to Section 10. Thus, the model's accuracy appears not to limit its capacity to capture a semantic gap between the article's Title and its content. This provides further evidence supporting our hypothesis, though it does not yet constitute a definitive or mathematical proof. We report the results on the MIND dataset. We will plot together the results for the JS and KT correlation scores as the results produced using the 2 LMs are identical.



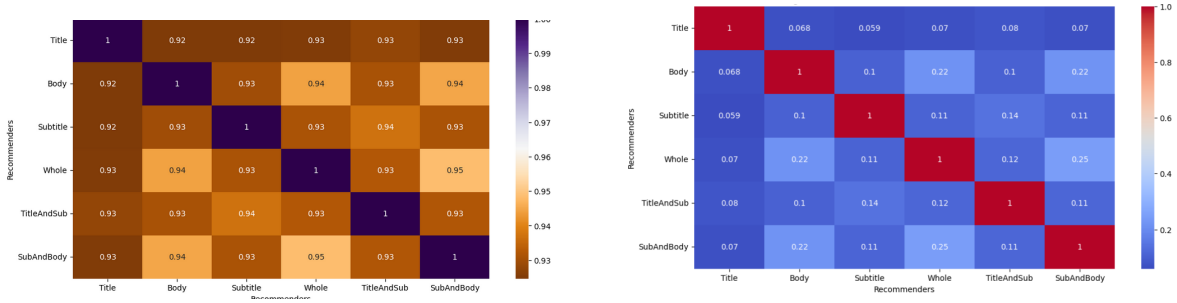
(a) JS (b) KT

Figure 18: Analysis for correct recommendadtions only on MIND for PureCB-KNN Recommender recommender.

Figure 18 shows some differences in the results depicted in Figure 6.KT results are similar overall, but JS values diverge notably when only correct recommendations are considered,suggesting that, although correct recommendations often feature the same articles across different models, the ranking of these items within the lists varies significantly. In other words, even when the recommenders agree on which articles to suggest, they differ in the priority assigned to each item.

11.2. FM Recommender

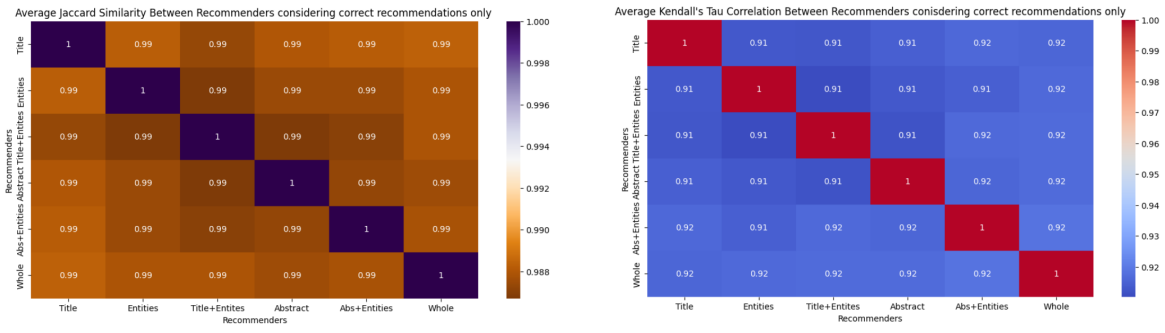
As done in Section 11.1 we omit the results for the different LMs if they are exactly the same.



(a) JS (b) KT

Figure 19: Analysis for correct recommendadtions only on EbNeRD for FM Recommender.

The results depicted in Figure 19 are completely coherent with those in Figure 8 and Figure 9. While the KT values are basically the same, as happened for PureCB-KNN Recommender the JS is generally higher. However, it is still clear how the Title is less correlated. In Figure 20 the MIND dataset results are depicted.



(a) JS (b) KT

Figure 20: Analysis for correct recommendations only on MIND for FM Recommender.

Figure 20confirms that the tests conducted on MIND are different from the ones obtained in EbNeRD (Figure

19). The JS scores are higher and there is less difference among the articles' parts with respect to Figure 19; also, the ranking measures denote a difference that is less evident than in EbNeRD. In all RS we note some misalignment in the results obtained over the two datasets.
In this Section we proved that, considering only correct recommendations, the results are coherent to those that consider all the impressions. Therefore, the suboptimal accuracy of the models does not render our intuition on using LMs for *clickbait detection* invalid. We gathered more evidence about a difference in the results on the two datasets that is probably due to inherent misalignment between them, which could require a deeper inspection.

12. Analysis on the Relationship between the Correlation Scores and Article Components' Lengths

In this section, we investigate whether the lengths of the article parts encoded could impact the results of Section 10. Specifically, the character count of each article part might be significant; for example, while a concept might be briefly presented in the Title, it is more likely to be elaborated upon in the Body, given the Body's generally greater length. Differences in length may lead to variations in embeddings, as longer sections like the Body and possibly Subtitles are more likely to share common terms and concepts related to the same topic. Such overlap could yield more similar embeddings across longer article parts, potentially affecting the recommendation outcomes. We consider only the relationships with the Title model. For each article part we computed its average length, then we checked if there was a relationship between the average lengths and the average correlation scores across the impressions (both JS and KT). As metrics we use the Pearson Correlation Coefficient [18] and the Spearman Rank Correlation Coefficient [53]. The Pearson Correlation measures the linear relationship between two continuous variables. Its values range from -1 (perfect negative correlation) to +1 (perfect positive correlation), with 0 indicating no linear relationship. The Spearman correlation assesses the monotonic relationship between two ranked variables. Like Pearson, its range is from -1 to +1.

12.1. PureCB-KNN Recommender

In this Section, we show the results of PureCB-KNN Recommender. We will first plot the similarities related the various article parts' lengths. Then the results for the Pearson and Spearman correlation scores. We start our analysis with the EbNeRD dataset.

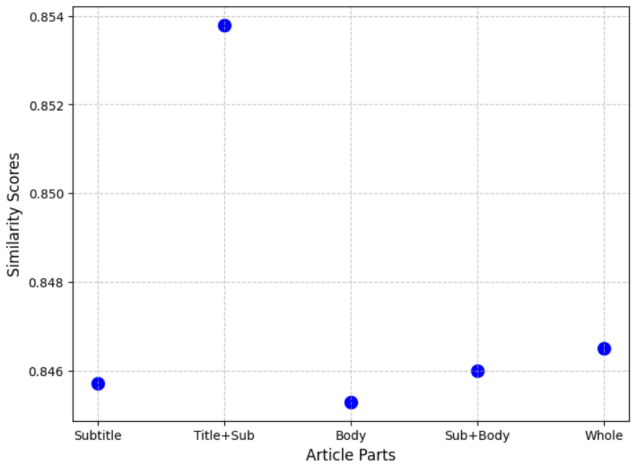


Figure 21: JS compared to average articles'parts lengths for PureCB-KNN Recommender (DistilBERT Embeddings)

Figure 21 plots the average JS of the Title embedding (y-axis) versus the other articles' parts average lengths(x-axis), the latter are presented in increasing order of length. The plot makes us understand that there is no linear relationship between the similarities and the lengths. To further explore this aspect, we calculate the Pearson and Spearman coefficients. We report only the results using DistilBERT since those we got with BERT are identical.

Table 7: JS Correlation Metrics with Title Model (DistilBERT)

Metric	Value
Pearson Correlation	-0.57
Spearman Correlation	0.2

Table 7 represents the Pearson and Spearman coefficients; this result indicates a moderate inverse relationship in a linear sense, but any rank-based or non-linear trend between the variables is weak and may not even be particularly meaningful. When studying KT correlations we saw results similar to those in Table **??**, so we will not be reporting them all. Next, we report the results obtained on MIND on the KT Correlation metric.

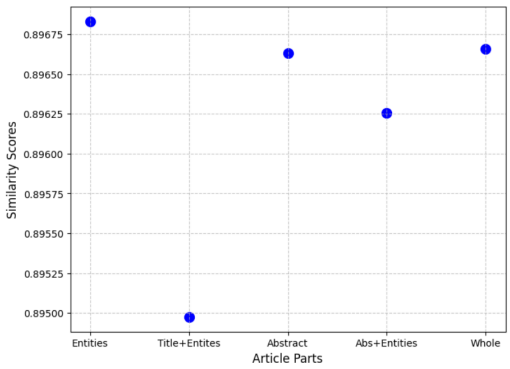


Figure 22: Analysis over KT on the MIND dataset: correlation vs parts' lengths

Table 8: KT Correlation Metrics with Title Model (DistilBERT)

Metric	Value
Pearson Correlation	-0.1069
Spearman Correlation	0.1

Figure 22 and Table **??**show no linear relationship. Data suggests that the variables are effectively uncorrelated, as neither a linear nor a monotonic relationship is meaningfully present.

12.2. FM Recommender

In this Section, we will analyze the results obtained using FM Recommender. We will plot one example per dataset: only considering DistilBERT since the ones with BERT are identical. We will plot the JS on the EbNeRD only since the results are mirrored in MIND.

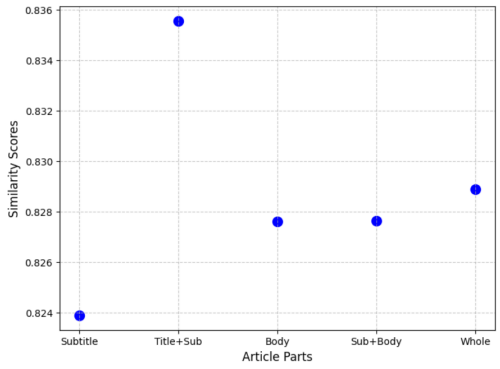


Figure 23: Analysis over JS on the EbNeRD dataset: correlation vs parts' lengths

Table 9: JS Correlation Metrics with Title Model (DistilBERT)

Metric	Value
Pearson Correlation	-0.2884
Spearman Correlation	0.4

In Figure 23 and Table 9 we report the results on the JS in the EbNeRD dataset. Again, the results show no linear no monotonic relationships between the similarities and the article parts' lengths.

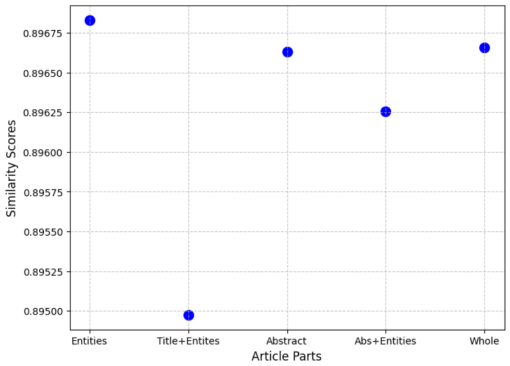


Figure 24: Analysis over KT on the MIND dataset: correlation vs parts' lengths

Table 10: KT Correlation Metrics with Title Model (DistilBERT)

Metric	Value
Pearson Correlation	-0.1069
Spearman Correlation	0.1

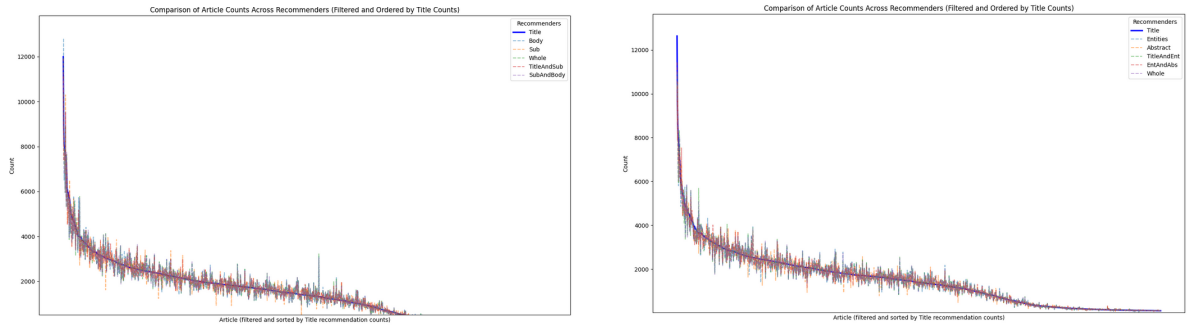
Also, for the MIND dataset and for the ranking metric there is no relationship as shown in Figure 24 and Table 10. In general, we note greater agreement between datasets and models both in the plots and in the correlation scores, with respect to Section 10 and Section 11.

13. How different Encoded Parts influence the Global Recommendation trends

In this Section, we inspect how the recommendation lists are influenced in a broader sense. We are interested in how many times the articles are recommended in the various scenario we studied in the previous Sections. To study this research question, we take the recommendation lists and count the occurrences of each article. Then, we plot for each article how many times it has been recommended by each RS. To ease the visualizations we filter out the 50% less viewed articles overall. Note that most of the articles appear less than 10 times,in the recommendation lists, therefore discarding them does not effect our study.

13.1. PureCB-KNN Recommender

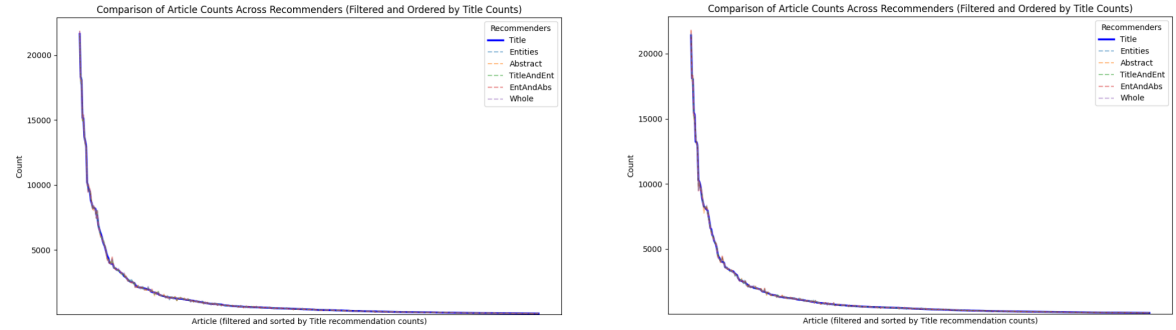
In figure 25 we report the results of PureCB-KNN Recommender on EbNeRD.



(a) DistilBERT (b) BERT

Figure 25: Comparison of Recommendation trends using different LMs using PureCB-KNN Recommender on EbNeRD

In figure 25 we note the differences in terms of overall count of recommendations for DistilBERT in 28a and BERT in 28b. In Section 11, we noted a difference in the recommendation lists for correct recommendations, also in this analysis we find that BERT embeddings have more accentuated peaks for some articles recommended by their title. We found therefore some peaks that could correspond to candidate *Clickbait* articles.
We report the results for PureCB-KNN Recommender over the MIND dataset in Figure 26.



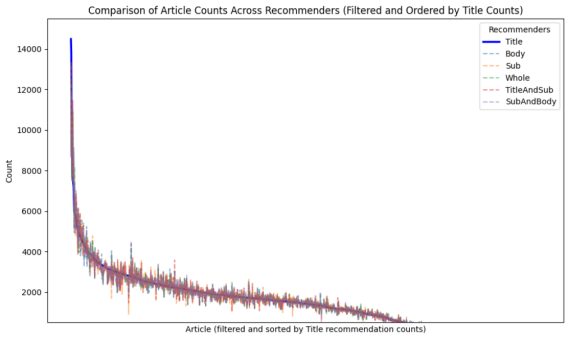
(a) MIND Articles Recommendation count with DistilBERT embeddings (b) MIND Articles Recommendation count with BERT embeddings

Figure 26

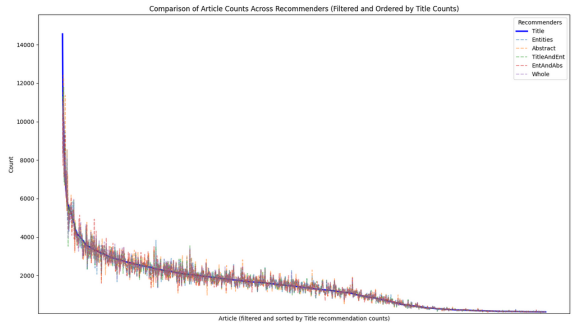
Figure 26 is quite different from 25. We note there far less peaks and, more importantly, less difference between the various models.

13.2. FM Recommender

In Figure 27 we report the results of FM Recommender over the EbNeRD dataset.



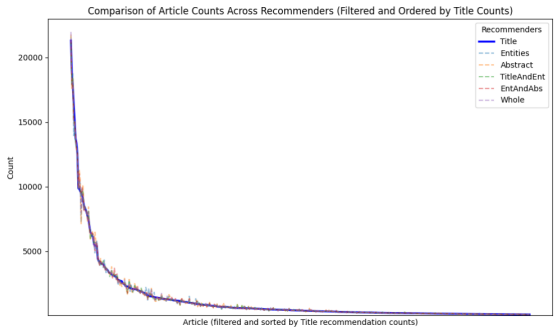
(a) EbNeRD Articles Recommendation count with DistilBERT embeddings



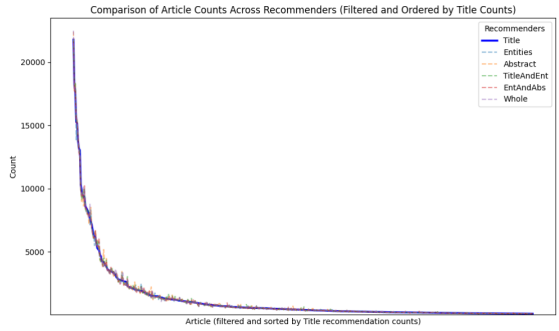
(b) EbNeRD Articles Recommendation count with BERT embeddings

Figure 27

Figure 27 reports results that are similar to those of figure 25. Peaks are higher in the latter, while the overall shape of the plot is similar."this Figure shows visible overlap between recommendation frequencies for every article.



(a) MIND Articles Recommendation count with DistilBERT embeddings



(b) MIND Articles Recommendation count with BERT embeddings

Figure 28

Figure 28 is consistent with figure 26, confirming key similarities across models. To begin with, there is not much difference between the results of BERT and DistilBERT embeddings, suggesting that the choice of these specific language models does not significantly affect the observed patterns in our recommendation lists. Additionally, as we observed in the EbNeRD dataset plots, there is only minimal variation in the results when transitioning between different recommendation algorithms, further supporting the stability of these patterns across model types.

In this section, our focus was a broad inspection of how recommendations change depending on which parts of the article are embedded, with the primary aim being to determine whether there are notable variations in the frequency with which certain articles appear in the recommendation lists, and, if so, to what extent. For the EbNeRD dataset, we observed that certain prominent peaks emerge specifically when using the Title recommender compared to others. This supports the notion that our approach for identifying potential *Clickbait* articles is not only sound but also effective in highlighting these differences. Consistent with the observations in earlier sections, we found that for the MIND dataset, the differences in results were less pronounced. This narrower margin of difference suggests that the variation between the title and other article sections may be somewhat dataset-dependent, providing a useful perspective on the applicability of our method across varied data sources.

14. Final Remarks

In this Thesis, we performed 4 experiments to check if LMs could be useful to detect semantic inconsistencies between the articles' Title and the actual content of the articles. In other words, we tried to perform *Clickbait* detection using LMs in an online setting. The results of the experiments we conducted confirmed that RSs that take as input the encoding of articles' title have a different behavior with respect to those that take as input

the encoding of the article content. We then checked if these results could have been caused by the incapability of the RS to provide correct recommendations, by running an analysis only on the *correct* recommendations. Subsequently, we checked if the lengths of the encoded parts influenced the results obtained. We did that by asserting if there was a linear or monotonic relationship between the correlation scores and the average lengths of encoded text in terms of characters. Again, the results of this Section confirmed that the results of Section 1 were due to a semantic gap between the article title and the article content. In the last experiment, we inspected how the recommendation trends changed globally for the articles, noting peaks for articles recommended by their Title. The last experiments denote an evident difference in behavior when RSs use Title information versus when they use the actual content. We believe that the articles that present an important difference in number of times they have been recommended when using the title encoding versus the content encoding, could be considered as candidate *clickbait* articles. When it comes to the datasets, we appreciated some difference when working with the MIND dataset. This can be due to a couple of reasons that both concern the inherent structure of the dataset. First, the MIND dataset did not provide actual body and subtitle for the articles, it provided an abstract which we reasonably considered as the content of the article, however it is not the same as having the full Body. The absence of the full Body information has certainly mitigated the differences in the recommendation lists. Secondly, there is a difference in the source of the articles, the topics treated in EbNeRD are generally lighter, referring to tabloid news and celebrity news (see Figure 1). Our hypothesis is that lighter news could be more prone to *Clickbait* phenomenon. Finally, MIND is a collection of articles that originate from different sources: such heterogeneity could have affected the results of our experiment and slightly differentiate them with respect to EbNeRD experiments' results.

15. Future Work

During the experiments we note that there was not much difference when it came to which LM was in use. Therefore, a possible next step could be the one to use more powerful LMs ,or even Large Language Models (LLMs), to verify if the results change incrementing the number of parameters. Also, a possible next step could involve more sophisticated RS for computing the recommendation lists, such as neural networks. Another possible next step would involve the use of more performing models in terms of AUC, even if in Section 11 we tried to demonstrate how our analysis is sound even for models that did not perform competitively in the Challenge. Another possible step forward could be to use a third dataset and verify if the results we got are confirmed: some possible choices may include the Adressa dataset [22] or the Globo.com dataset [11].

References

[1] Acheampong Francisca Adoma, Nunoo-Mensah Henry, and Wenyu Chen. Comparative analyses of bert, roberta, distilbert, and xlnet for text-based emotion recognition. In *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 117–121, 2020.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[3] Andrea Alari, Lorenzo Campana, Federico Giuseppe Ciliberto, Saverio Maggese, Carlo Sgaravatti, Francesco Zanella, Andrea Pisani, and Maurizio Ferrari Dacrema. Exploiting contextual normalizations and article endorsement for news recommendation. In *Proceedings of the Recommender Systems Challenge 2024*, RecSysChallenge '24, page 17–21, New York, NY, USA, 2024. Association for Computing Machinery.

[4] André Altmann, Laura Tolosi, Oliver Sander, and Thomas Lengauer. Permutation importance: a corrected feature importance measure. *Bioinform.*, 26(10):1340–1347, 2010.

[5] Paolo Basso, Arturo Benedetti, Nicola Cecere, Alessandro Maranelli, Salvatore Marragony, Samuele Peri, Andrea Riboni, Alessandro Verosimile, Davide Zanutto, and Maurizio Ferrari Dacrema. Pessimistic rescaling and distribution shift of boosting models for impression-aware online advertising recommendation. In *ACM RecSys Challenge 2023, Singapore, 19 September 2023*, pages 33–38. ACM, 2023.

[6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):993–1022, March 2003.

[7] Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. Learning to rank with nonsmooth cost functions. In Bernhard Schölkopf, John C. Platt, and Thomas Hofmann, editors, *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, pages 193–200. MIT Press, 2006.

[8] Kelsy Cabello-Solorzano, Isabela Ortigosa de Araujo, Marco Peña, Luís Correia, and Antonio J. Tallón-Ballesteros. The impact of data normalization on the accuracy of machine learning algorithms: a comparative analysis. In *International Conference on Soft Computing Models in Industrial and Environmental Applications*, pages 344–353. Springer, 2023.

[9] 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. 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*, pages 28–33. ACM, 2021.

[10] 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. Wide & deep learning for recommender systems. In Alexandros Karatzoglou, Balázs Hidasi, Domonkos Tikk, Oren Sar Shalom, Haggai Roitman, Bracha Shapira, and Lior Rokach, editors, *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016*, pages 7–10. ACM, 2016.

[11] Gabriel de Souza Pereira Moreira, Felipe Ferreira, and Adilson Marques da Cunha. News session-based recommendations using deep neural networks. In *Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems*, DLRS 2018, page 15–23, New York, NY, USA, 2018. Association for Computing Machinery.

[12] 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. Lightweight model for session-based recommender systems with seasonality information in the fashion domain. In *Proceedings of the Recommender Systems Challenge 2022*, RecSysChallenge '22, page 18–23, New York, NY, USA, 2022. Association for Computing Machinery.

[13] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, jan 2004.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[15] Nima Dolatnia and Alan Fern. Bayesian optimization with resource constraints and production. *Proceedings of the International Conference on Automated Planning and Scheduling*, 26:115–123, 03 2016.

[16] Nicolò Felicioni, Andrea Donati, Luca Conterio, Luca Bartoccioni, Davide Yi Xian Hu, Cesare Bernardis, and Maurizio Ferrari Dacrema. Multi-objective blended ensemble for highly imbalanced sequence aware tweet engagement prediction. In *Proceedings of the Recommender Systems Challenge 2020*, RecSysChallenge '20, page 29–33, New York, NY, USA, 2020. Association for Computing Machinery.

[17] Evelyn Fix and J. L. Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review / Revue Internationale de Statistique*, 57(3):238–247, 1989.

[18] David Freedman, Robert Pisani, and Roger Purves. Statistics (international student edition). *Pisani, R. Purves, 4th edn. WW Norton & Company, New York*, 2007.

[19] I. J. Good. Rational decisions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 14(1):107–114, 1952.

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

[21] Andrey Gulin, Igor Kuralenok, and Dmitry Pavlov. Winning the transfer learning track of yahoo!'s learning to rank challenge with yetirank. In Olivier Chapelle, Yi Chang, and Tie-Yan Liu, editors, *Proceedings of the Yahoo! Learning to Rank Challenge, held at ICML 2010, Haifa, Israel, June 25, 2010*, volume 14 of *JMLR Proceedings*, pages 63–76. JMLR.org, 2011.

[22] Jon Atle Gulla, Lemei Zhang, Peng Liu, Özlem Özgöbek, and Xiaomeng Su. The adressa dataset for news recommendation. In *Proceedings of the International Conference on Web Intelligence*, WI '17, page 1042–1048, New York, NY, USA, 2017. Association for Computing Machinery.

[23] John Hancock. Jaccard distance (jaccard index, jaccard similarity coefficient), 10 2004.

[24] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[25] Dietmar Jannach, Gabriel de Souza P. Moreira, and Even Oldridge. Why are deep learning models not consistently winning recommender systems competitions yet? a position paper. In *Proceedings of the Recommender Systems Challenge 2020*, RecSysChallenge '20, page 44–49, New York, NY, USA, 2020. Association for Computing Machinery.

[26] Manu Joseph and Harsh Raj. Gandalf: Gated adaptive network for deep automated learning of features, 2024.

[27] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3146–3154, 2017.

[28] Benjamin Kille, Frank Hopfgartner, Torben Brodt, and Tobias Heintz. The plista dataset. In *Proceedings of the 2013 International News Recommender Systems Workshop and Challenge*, NRS '13, page 16–23, New York, NY, USA, 2013. Association for Computing Machinery.

[29] Johannes Kruse, Kasper Lindskow, Saikishore Kalloori, Marco Polignano, Claudio Pomo, Abhishek Srivastava, Anshuk Uppal, Michael Riis Andersen, and Jes Frellsen. Eb-nerd a large-scale dataset for news recommendation. In *Proceedings of the Recommender Systems Challenge 2024*, RecSys Challenge '24, page 1–11. ACM, October 2024.

[30] Maciej Kula. Metadata embeddings for user and item cold-start recommendations, 2015.

[31] Charles X. Ling, Jin Huang, and Harry Zhang. Auc: a better measure than accuracy in comparing learning algorithms. In *Proceedings of the 16th Canadian Society for Computational Studies of Intelligence Conference on Advances in Artificial Intelligence*, AI'03, page 329–341, Berlin, Heidelberg, 2003. Springer-Verlag.

[32] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[33] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

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

[35] Jonas Mockus. *Bayesian Approach to Global Optimization*, volume 37, pages 473–481. 01 2006.

[36] Gabriel De Souza P. Moreira, Dietmar Jannach, and Adilson Marques Da Cunha. Contextual hybrid session-based news recommendation with recurrent neural networks. *IEEE Access*, 7:169185–169203, 2019.

[37] Liudmila Ostroumova Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6639–6649, 2018.

[38] Kanglin Qu, Jiucheng Xu, Qincheng Hou, Kangjian Qu, and Yuanhao Sun. Feature selection using information gain and decision information in neighborhood decision system. *Applied Soft Computing*, 136:110100, 2023.

[39] Steffen Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000, 2010.

[40] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

[41] Dalwinder Singh and Birmohan Singh. Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, 97:105524, 2020.

[42] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014.

[43] Alexei Stepanov. On the kendall correlation coefficient, 2015.

[44] Vladimir Vovk. The fundamental nature of the log loss function, 2015.

[45] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. *CoRR*, abs/1708.05123, 2017.

[46] Ruoxi Wang, Rakesh Shivanna, Derek Zhiyuan Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed H. Chi. DCN V2: improved deep & cross network and practical lessons for web-scale learning to rank systems. In Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia, editors, *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 1785–1797. ACM / IW3C2, 2021.

[47] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

[48] Zhiqiang Wang, Qingyun She, PengTao Zhang, and Junlin Zhang. Correct normalization matters: Understanding the effect of normalization on deep neural network models for click-through rate prediction, 2020.

[49] Vinicius Woloszyn, Henrique D. P. dos Santos, Leandro Krug Wives, and Karin Becker. Mrr: an unsupervised algorithm to rank reviews by relevance. In *Proceedings of the International Conference on Web Intelligence*, WI '17, page 877–883, New York, NY, USA, 2017. Association for Computing Machinery.

[50] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[51] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 3597–3606, 2020.

[52] In-Kwon Yeo and Richard A. Johnson. A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4):954–959, 2000.

[53] Jerrold H Zar. Spearman rank correlation. *Encyclopedia of Biostatistics*, 7, 2005.