A Clustering Based Article Template Recommendation System for Newspaper Editors

Sebastián Gallardo, Bruno Génuit, Dorian Mazauric, Pierre Kornprobst

A Clustering Based Article Template Recommendation System for Newspaper Editors

Sebastián Gallardo , Bruno Génuit, Dorian Mazauric, Pierre Kornprobst

RESEARCH REPORT

N° 9560

November 2024

Project-Team Biovision

ISSN 0249-6399 ISRN INRIA/RR--9560--FR+ENG

A Clustering Based Article Template Recommendation System for Newspaper Editors

Sebastián Gallardo* †, Bruno Génuit†, Dorian Mazauric*, Pierre Kornprobst*

Project-Team Biovision

Research Report n° 9560 — November 2024 — 33 pages

* Université Côte d'Azur, Inria
† Demain un Autre Jour

RESEARCH CENTRE
Centre Inria d'Université Côte d'Azur
2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Abstract: Newspaper editors face significant challenges in page production, as printed newspapers are becoming less profitable and more costly to produce. Improving productivity in this process is crucial. In this paper, we focus on the task of selecting the most appropriate article template for given content—a process that is time-consuming and difficult. We demonstrate how techniques from Recommendation Systems (RS) can be adapted and extended to assist newspaper editors by recommending the best templates based on their needs and preferences. We propose a clustering-based recommendation system that promotes diversity, which is a critical requirement in this context. Our method consists of two phases: first, an offline clustering phase that uses a graph-matching neural network to compute a custom similarity measure (*GMN*) between templates, modeled as graphs. This clustering process is based on content and structural information, independent of user preferences. The second phase is an online recommendation system that incorporates both user preferences and content requirements, promoting templates that minimize similarity. Compared to existing methods, our approach outperforms in terms of novelty and diversity, especially in cases where high-rated items are clustered together. We present promising results through use case examples, demonstrating the practical applicability of our method for real-world scenarios, while also opening avenues for future studies with industry professionals. While this study provides a first step towards applying recommendation systems to optimize the selection of article templates in the newspaper production process, further research is needed to extend this work to full-page layouts and multi-page designs, incorporating more comprehensive recommendations that address the broader scope of newspaper production.

Key-words: clustering-based recommendation, graph neural networks, newspaper layout, journalism, diversity

Système de recommandation de modèles d'articles basé sur le clustering pour les éditeurs de journaux

Résumé : Les éditeurs de journaux font face à des défis majeurs dans la production de pages, à mesure que les journaux imprimés deviennent moins rentables et plus coûteux à produire. Améliorer la productivité de ce processus est donc essentiel. Dans cet article, nous nous concentrons sur la tâche de sélection du modèle d'article le plus approprié pour un contenu donné—un processus chronophage et complexe. Nous montrons comment les techniques des systèmes de recommandation (RS) peuvent être adaptées et étendues pour assister les éditeurs en leur proposant les meilleurs modèles en fonction de leurs besoins et préférences. Nous proposons un système de recommandation basé sur le clustering qui favorise la diversité, une exigence cruciale dans ce contexte. Notre méthode se compose de deux phases : d'abord, une phase de clustering hors ligne utilisant un réseau neuronal de correspondance de graphes pour calculer une mesure de similarité personnalisée (*GMN*) entre les modèles, modélisés comme des graphes. Ce processus de clustering repose sur des informations structurelles et de contenu, indépendamment des préférences des utilisateurs. La deuxième phase est un système de recommandation en ligne qui intègre à la fois les préférences des utilisateurs et les exigences liées au contenu, en favorisant les modèles minimisant la similarité. Comparée aux méthodes existantes, notre approche se distingue par sa capacité à améliorer la nouveauté et la diversité, notamment dans les cas où les articles les mieux notés sont regroupés dans des clusters. Nous présentons des résultats prometteurs à travers des exemples d'utilisation, démontrant l'applicabilité pratique de notre méthode dans des scénarios réels, tout en ouvrant des perspectives pour des études futures avec des professionnels du secteur. Bien que cette étude constitue une première étape vers l'application des systèmes de recommandation pour optimiser la sélection de modèles d'articles dans le processus de production de journaux, des recherches supplémentaires sont nécessaires pour étendre ce travail à des mises en page complètes et à des designs multi-pages, en proposant des recommandations plus complètes pour répondre aux besoins élargis de la production journalistique.

Mots-clés : Recommandation basée sur le clustering, Réseaux neuronaux basés sur des graphes, mise en page de journaux, journalisme, diversité

Contents

1 Introduction

Before the advent of the Internet, print journalism production was tightly coupled with a specific media format: the newspaper page itself. Journalists wrote articles directly in layouts defined by the physical space constraints of the paper, known as a "layout-driven" approach. This process not only limited the volume of articles due to space constraints but also guided the writing process according to a predefined visual structure [7].

The digital era has transformed news production. Journalists now write for web platforms first, without the rigid layout constraints of print, in what can be considered a "content-driven" workflow. This shift has drastically increased the volume of articles produced and the speed at which they must be published. However, it has also introduced new challenges for the print version, which is still a matter of interest for big publishers [11]. While the focus on web content has increased, the economic pressures on print media have intensified, making it essential to

deliver high-quality articles in formats that remain engaging and visually accessible [25, 11]. This change has heightened the need for effective content management systems (CMS), which can help publishers efficiently make print editions [38].

A practical example of this need, drawn directly from our collaboration with an industrial partner, can be illustrated through the following user scenario: "As a journalist, I want to quickly select the best template for the article I just wrote so that I can put it into production efficiently." In typical CMS workflows, journalists must not only write articles but also determine their final layout within the system, choosing from a catalog of templates that potentially includes hundreds of options. These templates are complex, representing diverse combinations of content regions—each shaped as a polygon that can vary in size, structure, and purpose (e.g., text, images, titles). CMS platforms offer filtering features, but the results can remain overwhelming, with dozens of similar layouts complicating the selection process.

To address this challenge, we propose leveraging recommendation system techniques [6] to assist journalists in selecting article templates. Specifically, following the use case defined by our industrial partner, once a near-final draft of the article is ready at the stage when layout decisions come to the forefront—the system should offer a tailored set of template recommendations. These recommendations would be not only relevant to the article's content but also diverse in structure. In doing so, the recommendation process would enhance both the efficiency and quality of template selection, enabling journalists to quickly explore a curated selection of optimal layouts rather than sifting through an unsorted, extensive list of templates

Our proposed recommendation system (TRS) is designed to address two critical factors in template selection: diversity and novelty. Traditional recommendation methods often rely on user feedback and similarity measures based on ratings [37]; however, this approach does not align well with our needs for two reasons. First, template relevance varies contextually, depending on the specifics of the article, and usage patterns alone do not capture this. Second, template layouts are inherently structured as two-dimensional designs, requiring a layout-aware similarity measure. Therefore, we incorporate a graph-matching neural network to evaluate structural layout similarity and clustering templates based on their design characteristics rather than user popularity alone. This approach provides diversity by varying layout structures within recommendations and supports novelty by selecting templates that are underutilized yet relevant.

Interestingly, this new application raises two research questions. The first question explores how a structurally-based clustering approach can leverage layout similarities to enhance diversity and novelty when items are spatial compositions. The second question evaluates our system's performance relative to other clustering-based recommendation systems, particularly in 'long-tail' usage distributions [29] where high-usage templates are clustered rather than dispersed across clusters.

More specifically, our contributions to this work are:

- A recommendation framework focused on template selection in CMS environments, extending recommender system applications to new areas within journalism and publishing.
- A novel clustering-based approach that maximizes novelty and diversity while preserving layout compatibility with the written content.
- The application of a graph-matching neural network specifically trained to assess structural similarity between templates, enabling layout-aware recommendations.
- The release of a new dataset of real newspaper templates that offers a valuable resource for recommender system research in the publishing domain.

This paper is organized as follows. In Sec. 2, we briefly review the state-of-the-art In Sec. 3, we present our recommendation system, In Sec. 4, we show results and discuss some key features

of our approach. In Sec. 5, we will discuss some essential aspects of our method, notably its generalization power and applicability. Finally, we conclude in Sec. 6.

2 Related work

News recommendation systems (see survey [30])

Recommendation systems (RS) arises as an effective solution to manage the increasing amount of multi-modal content. As explored in the survey presented in [10], today's recommender systems suggest items of various media types, including audio, text, visual (images) and videos, considering both ratings and latent information of these different types of media.

Regarding the news industry, which is the main application field of this paper, news recommender systems (NRS) has been increasingly used to provide better suggestions of online news (i.e. news content) from different providers [30]. However, as stated by [4], standard accuracy values used in RS validation might be not enough to account for RS quality in news recommendations. In fact, [4] discuss about "beyond accuracy" metrics like diversity or novelty, and then extended the focus to "value-aware RS", where business value is taken into account. The authors conclude that value-aware RS is still an under-researched area.

Regarding newspaper layouts, in [14] a RS to propose layouts for automated design of magazine covers to amateur designers is presented, based on user preferences and high level features like "mood" (i.e. formal, clean or dynamic). Also, in [8], a RS that uses reinforcement learning method known as multi-armed bandit is presented, to solve marketing email layout recommendation problem. Our focus is on newspaper layouts recommendation but measuring similarity based on its structure, and considering diversity mechanisms to avoid proposing a set of too similar layouts.

Long-Tail Recommendation methods (see survey [29])

In short-head/long-tail distributions, or simply long-tail distributions, most of the items has lower rating values than a small subset of items that concentrates most of the usage (coined *short head*). This supposes a problem because unpopular items might be ignored by certain recommendation systems. Taking into account this items is closely related to the diversity problem. Qin [29] summarizes the recent challenges and solutions in this type of data distributions. Previous work in long-tail recommendation methods began with clustering-based methods, like Park and Tuzhilin [27] that studied the effect of applying data mining methods to predict rating values on each individual item in the dataset (*Each Item* method) in comparison to cluster the entire dataset and predict rating values of each group instead (*Total Clustering*). Finally, they proposed a solution that only clusters the tail of the distribution, and uses *Each Item* method for the short-head, which improves error rate of the predictions and is less computationally expensive. In addition, Park [26] improves its own work by proposing an adaptive clustering tail method that determine automatically head and tail bounds and cluster items according to their number of ratings, instead of finding splitting points manually like the previous version. Also, other type of RS were proposed, from graph-based methods [39, 16, 24] to deep learning approaches [33, 32, 22, 3]. It is important to note that we focus our analysis not only in clustering-based methods in long-tail distributions, but also in an interesting case when short-head items are clustered together in a small subset of clusters: in this case, choosing how many templates to recommend per cluster is not a trivial task since popularity-based strategies will have an important bias. We explore deeply this concern in the corresponding section in this work.

Diversity and Novelty in RS (see survey [18])

Diversity has become one of the leading topics of recommender system research. To start, even if it is a known concept, its conceptualization is ambiguous; Vrijenhoek et al. [37] provide an overview of goals and relevant aspects related to diversity of three different public service media organizations, concluding that conceptualization of diversity greatly varies, and it is unlikely that a standardized conceptualization will be achieved. Instead, cases should be analyzed one-by-one.

Regarding techniques, there are several ways to tackle diversity, like combining different item ranking techniques that offer diversity [1], directly modeling this as a binary optimization problem between similarity and diversity [13], or even treating it as a multi-objective that considers many variables in a way to maximize both accuracy and diversity [31]. However, the clustering-based recommendation is a simpler and more intuitive way to ensure diversity (see survey [5]), where items are first clustered, and then recommendations are built by selecting items from different clusters. This method maximizes diversity while preserving similarity, which is exactly what we seek in our approach. Among existing clustering-based methods, here we started from the method called ClusDiv [2], which focuses on maximizing diversity in top-N recommendation systems. One of the main advantages of this method is that it allows users to customize the diversity level of their recommendations with a very low time complexity [5]. However, we consider that clustering based on ratings, as ClusDiv does by default, is not aligned with diversity based on content (layouts in this work) similarity, but with what some authors named *novelty*: this concept measures how novel the recommendations are, considering item's popularity, to deal with the popularity-bias present in some RS. We think that a method that clusters based on the size of the recommendation list and user ratings not necessarily promotes diversity, but novelty instead, and that is why we propose in this work a clustering-based method that promotes both diversity and novelty as the main focus.

Graph neural networks (GNN) (see survey [12])

Graph neural networks allow the generation of vector representations of nodes and edges to be used to asses graph differences. In recommendation system literature, there are many works that implement this by finding graph representations of the relationships between users and items in the system and then applying GNN to learn latent vector representations, in several steps of a recommendation system [15, 21, 40, 20, 23]. This is a very useful approach since the relationships between users and items can be very complex. Also, the powerful graph-matching architecture proposed by [19], a novel graph neural network approach that considers cross-graph communication between a pair of graphs to compare, was applied in the Collaborative Filtering architecture, as proposed by Su, Zhang et al. [35]. In our work, we propose to use this graph-matching neural architecture to model templates as graphs and produce a similarity measure between them. With this approach, template inner structure and layout can be used to produce clusters in our clustering-based recommendation method.

3 Method

3.1 Method Overview

In this paper, we propose a clustering-based template recommendation method. The input to the method consists of the following:

- A catalog of templates $\mathcal{T}$, where templates are essentially rectangular layouts made up of different elements. Each element occupies a region, which can take the form of any polygon

(concave or convex, with or without holes), and may represent different content classes (see Fig. 1).

- Preference history, referred to as *template usage*, which tracks the use of each template in newspaper pages produced over a specific period. If a template from the catalog was not used during this period, we assign it a usage value of zero. Unlike other contexts where unrated might imply "unknown," here it clearly indicates no usage.
- Requirements derived directly from the article's content, which define a subset of admissible templates that satisfy those requirements.
- The number of suggested templates (N) we want to generate as the final output.

Our method has two main stages:

- Offline phase: In this stage, we handle diversity through clustering, which is performed using a graph-matching similarity measure.
- Online phase: This stage generates recommendations by selecting templates from different clusters. The final output is a list of the top-N recommended templates.

These two stages are presented in greater detail below.

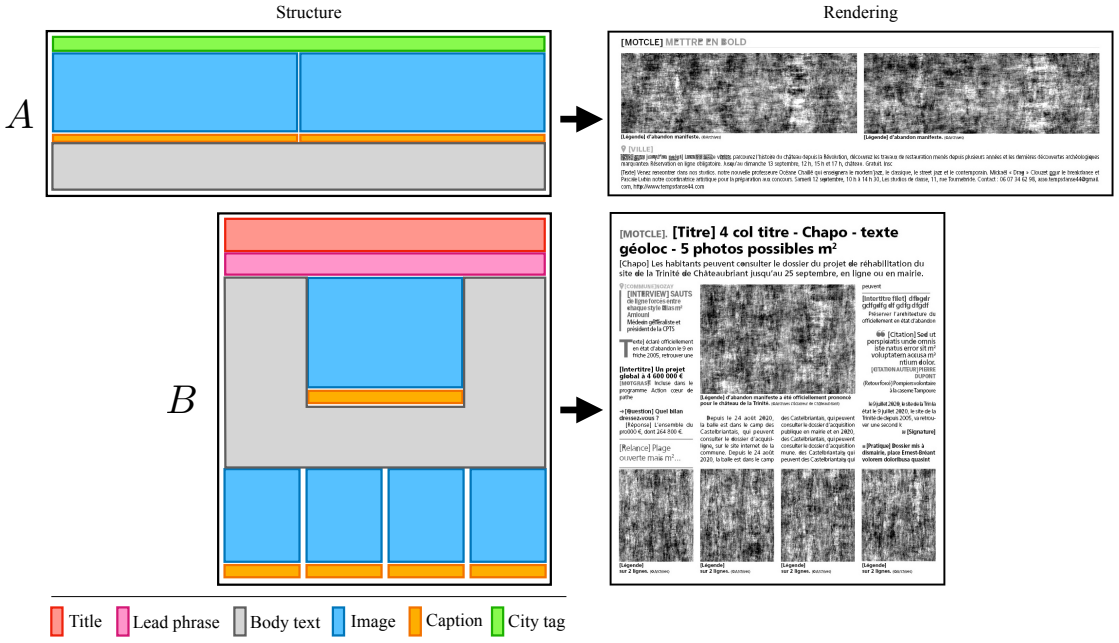

Figure 1: Two examples of templates with their abstract structure composed of different elements (left) and a corresponding realistic rendering (right).

3.2 Offline Phase: Clustering Templates Based on Similarity Measure

In this phase, we train a neural network to learn the similarities between templates. We then use this trained network to generate a distance matrix for the entire dataset, which is crucial for clustering the templates. These steps are performed a priori, meaning they occur before any recommendation queries and are not conducted in real-time.

***GMN* Similarity Measure.** To cluster templates, we propose a similarity measure that takes into account the inner structure of the templates. Our goal is to adopt an approach that captures the structural aspects of templates, moving beyond a purely positional similarity, such as that provided by the Intersection-over-Union (also known as the Jaccard index [9]).

To achieve this, we trained the graph-matching neural network (GMN) introduced in [28]. This network, referred to as LayoutGMN, learns a similarity metric by modeling pairs of templates as graphs and computing an embedding vector for each template. The process also incorporates attention-based weights to facilitate effective cross-graph communication. Subsequently, we can compute a negative-signed standard Euclidean distance between the two embedding vectors, yielding our *GMN*-based similarity measure.

Unfortunately, since this is a supervised approach, one of the problems is the absence of labeled data. To solve it, the authors in [28] weakly labeled data using standard IoU, which has several problems as acknowledged in their work, like (1) sensitivity to elements' slight misalignments between two similar templates, or (2) concerns for optimization problems, because if there is no overlapping between elements, IoU is zero and there is no gradient to optimize, so the optimization algorithm does not know in which direction the gradient should improve.

In our case, to label the data, we proposed an adaptation of IoU for structured layouts called template-based similarity measure IoU (denoted by *tdIoU*), which is based on dIoU [41] but tailored for our specific problem (see Appendix for technical details).

After training of GMN using $tdIoU$-labeled data, we obtain a resulting *GMN* similarity measure denoted by $GMN(.)$ that can be applied to measure similarity of two templates t_1 and t_2. $GMN(.)$ is always negative but not lower bounded, where $GMN(t_1, t_2) = 0$ means $t_1 = t_2$. Given this, we defined the following distance measure to do the clustering:

$$d(t_1, t_2) = |GMN(t_1, t_2)|. \tag{1}$$

Clustering Method We employ agglomerative clustering based on a clustering threshold value $\mathcal{C}_{th}$. This method begins with each template isolated in different clusters and recursively merges pairs of clusters when the average distance $d_{k=1,2}$ between clusters is below $\mathcal{C}_{th}$. Unlike other clustering-based recommendation systems, such as ClusDiv [2], the number of clusters is not determined by the size N of the recommended list. Instead, it relies solely on the independent tunable parameter $\mathcal{C}_{th}$, which depends on the desired tolerance for structural similarity of templates in the dataset.

3.3 Online Phase: Template Recommendation System (TRS)

In this phase, we have the set of M clusters associated with the set of templates $\mathcal{T}$ already computed in the offline phase of our method.

Let us describe each step of our method incrementally, with reference to the pseudocode presented in Algorithm 1 and illustrated in Fig. 2.

Define rating values (Alg. 1, lines 1–3) We begin by assigning rating values of each template to be equal to its corresponding usage.

Filter admissible templates (Alg. 1, line 4) We filter the set of templates $\mathcal{T}$ based on the requirements (e.g., templates with a title, or those with two images and a title). This results in a set of *admissible templates* $\mathcal{T}^A$, which will be used in the recommendation process.

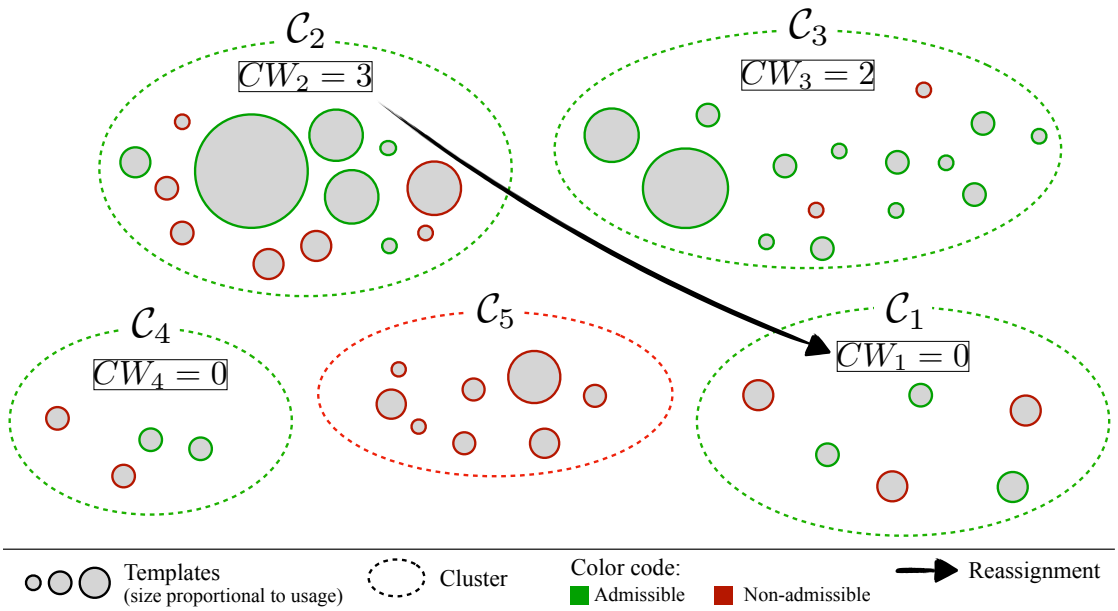

Figure 2: Illustration of our TRS algorithm under the assumption that $CW_{th} < 3$. Each template is represented by a grey circle, with size proportional to its relative usage; templates are grouped into clusters outlined with dotted lines. Based on defined requirements, admissible templates are marked with green borders, while non-admissible templates have red borders. Cluster boundaries are also color-coded: clusters with at least one admissible template are marked in green, while those without (such as $\mathcal{C}_5$) are marked in red. Using initial cluster weights, the reassignment algorithm first redirects weight values to $\mathcal{C}_2$ until either CW_2 reaches the threshold CW_{th} or the maximum number of admissible templates in $\mathcal{C}_2$ is reached. Subsequently, weights are also reassigned to $\mathcal{C}_1$, given that it has the lowest CW and the maximum distance from $\mathcal{C}_2$. The final phase of the algorithm corresponds to the selection step, which recommends CW_i templates per cluster $\mathcal{C}_i$, prioritizing templates with the highest rating values (larger green circles) and applying a proximity penalty to reduce similarity among selected templates and further enhance diversity.

Algorithm 1 Template Recommendation System (TRS) algorithm

Require: N: size of recommendation list, CW_{th}: diversity threshold, $\mathcal{T}$: list of templates,$\{C_i\}$: list of clusters, usage of each template in $\mathcal{T}$, requirements.

for all templates $t_{i,k}$ in each cluster $\mathcal{C}_i$ **do**
  assign rating value $r_{i,k}$ equal to usage of $t_{i,k}$
end for
$\mathcal{T}^A \leftarrow$ filtered $\mathcal{T}$ based on requirements, sorted by $r_{i,k}$ in decreasing order.
Top-$N \leftarrow N$ templates in $\mathcal{T}^A$ with highest $r_{i,k}$.
for $n = 1, ..., N$ **do**
  $i \leftarrow$ cluster that contains template n in the Top-N list.
  $CW_i = CW_i + 1$
end for
for `each source cluster` $\mathcal{C}_i$ **do**
  $\mathcal{T}_i^A \leftarrow$ set of admissible templates in cluster i
  while $CW_i > \min\{|\mathcal{T}_i^A|, CW^{th}\}$ **do** ▷ Reassign CW_i until reach threshold and $|\mathcal{T}_i^A|$
    $M = \{C_m\} \leftarrow$ set of clusters with $CW_m = \min_k CW_k$
    $C_j \leftarrow$ cluster with $max_{C_m \in M} \min_{t_{i,k} \in \mathcal{T}_i^A, t_{j,l} \in \mathcal{T}_j^A} GMN(t_{i,k}, t_{j,l})$
    $CW_j = CW_j + 1$
    $CW_i = CW_i - 1$
  end while
end for
while $CW_i > 0$ for any $\mathcal{C}_i$ **do**
  $t_{i,k} \leftarrow$ best template in $\mathcal{T}^A$ in terms of $r_{i,k}$
  if $CW_i > 0$ **then**
    add $t_{i,k}$ to output list
    $CW_i = CW_i - 1$
    Remove $t_{i,k}$ from $\mathcal{T}^A$
    for all $t_{i,l}$ where $GMN(t_{i,k}, t_{i,l}) < \gamma$ **do**
      $r_{i,l} \leftarrow r_{i,l}/\kappa$ ▷ κ should be $\sim \infty$
    end for
    Re-sort $\mathcal{T}^A$ in decreasing order of rating values $r_{i,l}$.
  else
    Remove $t_{i,k}$ from $\mathcal{T}^A$
  end if
end while
Return: output list.

Initialize the CW algorithm (Alg. 1, lines 5–9) The goal here is to determine how many templates to propose from each cluster. Given clusters, following the notation in [2], we refer to these numbers as *cluster weight* (CW), denoted by $\{CW_i\}_{\{1,...,M\}}$. Note that, unlike the classic ClusDiv, the number of clusters does not depend on N. However, we maintain the constraint $\sum_i CW_i = N$ to ensure that exactly N templates will be recommended. We assume that at least one admissible template exists in the dataset; otherwise, the system will simply return an empty set.

The method works as follows: using the parameter N, we extract the top-N templates with the highest rating value in $\mathcal{T}^A$. We then initialize $CW_i = 0$ and iteratively increase CW_i for the cluster $\mathcal{C}_i$ that contains items from the top-N list of templates.

Reassign CW_i weights (Alg. 1, lines 10–18) We reassign some CW_i values to increase the diversity of the recommendations by exploring underutilized clusters, up to a maximum number of templates per cluster defined by the threshold value CW^{th}. To achieve this, we adapt the original CW algorithm defined in [2] to consider only admissible templates. A key assumption in Algorithm 1 is that parameter N is smaller than $\sum_j |\mathcal{T}_j^A|$, i.e., the total number of admissible templates for the specific query. If not, the algorithm simply outputs all admissible templates. A critical feature of this algorithm is the selection process for increasing CW_j in cluster j from a source cluster i. Here we prioritize clusters with the smallest CW_j that contain admissible templates, and we select the farthest cluster j based on the distance function $d(.)$ powered by GMN. This distance is determined by calculating the minimum distance between all admissible templates in the source cluster i and those in cluster j. This procedure ensures increased diversity in the recommendations by considering the greatest distance between clusters.

Select templates for recommendation (Alg. 1, lines 19–31) Finally, we select templates from each cluster based on their corresponding CW_i values. The algorithm identifies the best template t_j from $\mathcal{T}^A$ in terms of rating value and checks whether it belongs to a cluster with $CW_i > 0$. If so, we add it to the recommendation list and decrease the corresponding CW_i by one. To further enhance diversity, we introduce a local suppression mechanism, which involves reducing the rating values of all templates similar to the selected one. Specifically, for templates t_k where $GMN(t_j, t_k) < \gamma$, their rating values are reduced by a factor κ, which has a very big value to strongly inhibit similar templates. The parameter γ is tunable ($\gamma < H$). This acts as a "soft removal", meaning that while we avoid recommending templates that are too similar to those already selected, we do not fully eliminate them from the candidate pool.

At the end of this final step, we obtain a recommendation list $L^N = [t_{k_1,i_1}, t_{k_2,i_2}, ..., t_{k_N,i_N}]$ with improved diversity and novelty.

4 Results

4.1 Dataset Characteristics

Catalog of Templates $\mathcal{T}$

We curated a dataset $\mathcal{T}$ of 452 templates from a real newspaper, which we processed to ensure it is clean and shareable. Each template is represented by (1) a JSON file defining the layout, where each element is characterized by its class and geometry, and (2) a pre-visualization with randomly generated text for each element, with image areas replaced by content whose phase is randomized (see Fig. 1).

For reference, some general information about the dataset is provided in Table 1, along with histograms of the number of columns, images, and aspect ratios in Fig. 3. These visualizations highlight the diversity of the dataset in terms of shapes and elements.

Then, key statistics about template usage are presented also in Table 1. Notably, only a few templates show very high usage, while 17% of templates have zero usage. Additionally, over 90% of templates have been used fewer than 1,600 times, and 96% have been used fewer than 8,000 times. This indicates a long-tail distribution, which, as discussed in [29], can influence both the quality and diversity of the recommendation system.

$\mathcal{T}$ together with CSVs files for template usage is publicly available at the following page: `https://team.inria.fr/biovision/trs`.

Description	Value
Number of templates	452
Average width (mm)	137.1
Average height (mm)	132.9
Average number of images	1.48
Average number of different classes	3.29
Average number of columns	3.19
Template ID with max usage	3142
Max usage	38005
Number of Zero-usage templates	79
Average usage	781.8
Usage deviation	3433.4
% templates with usage < 1600	92%
% templates with usage < 8000	96%

Table 1: Table with general information about $\mathcal{T}$ dataset.

4.2 Offline Phase Results

First, we focus on the training of the LayoutGMN network. To generate the training dataset, we create an augmented version of $\mathcal{T}$, denoted as $\mathcal{T}^+$. For each template containing images, we produce all possible variations by removing $i = 0, 1, ..., n$ images, thereby creating more space for body text. This augmentation process results in a dataset $\mathcal{T}^+$ with 5,341 templates. Using $\mathcal{T}^+$, we trained Layout GMN using the parameters listed in Table 2. The training followed the triplet framework detailed in [28], ensuring that no template was repeated between the training and test sets for any triplet. With these parameters, the model achieved 98% accuracy on the test data, comparable to the results reported by [28].

Next, we proceed with the analysis of the clustering step. Using the hand-tuned threshold specified in Table 2 for agglomerative clustering, we obtained 81 clusters. Figure 4 presents the distribution of these clusters based on the number of templates and their usage. Notably, we identified over 20 isolated templates, which are designed for specific purposes and have unique structures, as well as low usage, as shown in Fig. 4 (top). Additionally, Fig. 4 (bottom), highlights two large clusters in terms of usage, which primarily consist of general-purpose templates, such as those containing only body text, a single image, or standard image-with-caption templates of varying sizes.

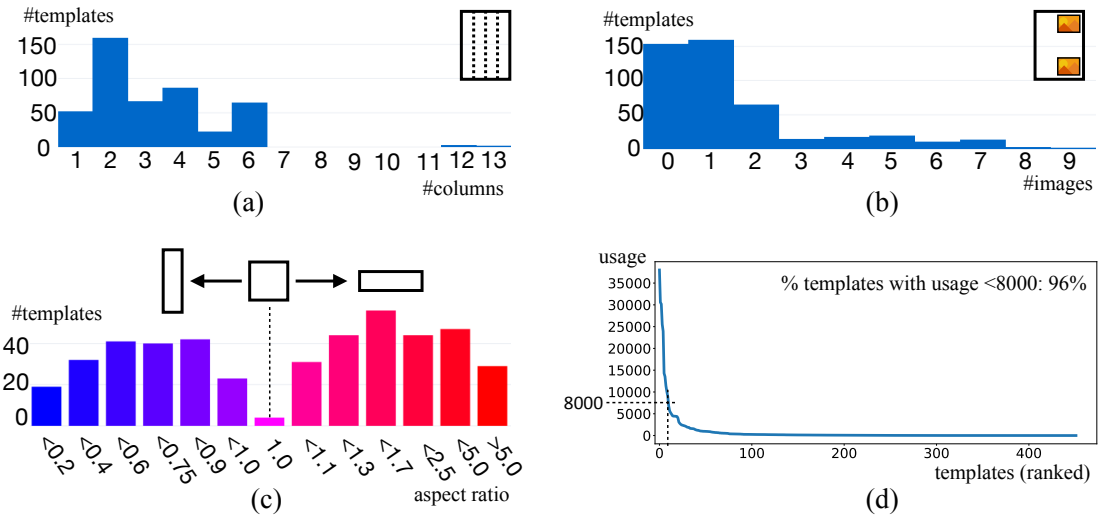

Figure 3: Overview of template dataset characteristics: (a) distribution of column counts per template, (b) distribution of image counts within templates, (c) distribution of template aspect ratios, and (d) usage distribution of templates, reordered by decreasing usage. These plots provide a general overview of the structural and usage diversity in the dataset.

Description	Value
Layout *GMN* training parameters	
Batch size	20
Epochs	500
Dataset size	5341
Training set size	80%
Validation set size	20%
Clustering parameters	
$\mathcal{C}_{th}$	2.8
Online phase parameters	
γ	0.5
κ	10^7

Table 2: Main parameters used for *GMN* clustering.

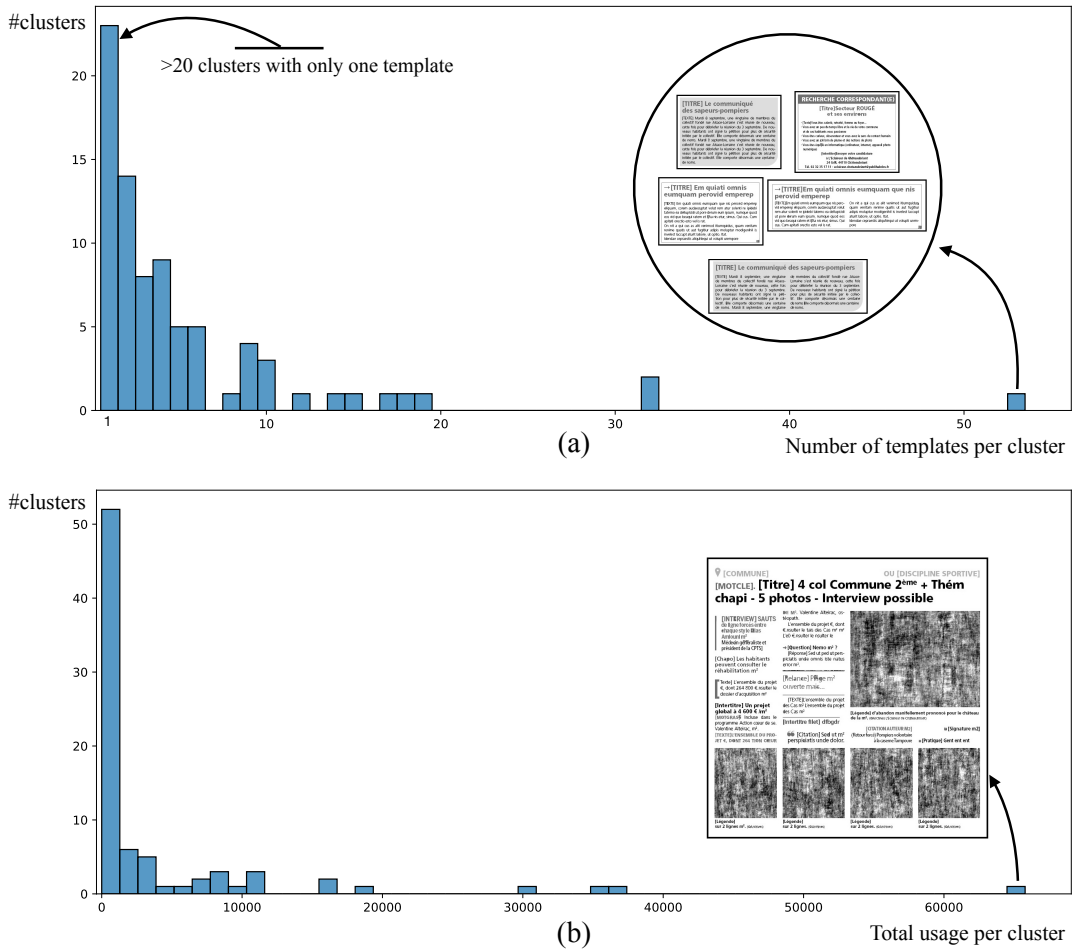

Figure 4: Cluster analytics: (a) Histogram of the number of templates per cluster. The histogram reveals over 20 clusters containing a single template, alongside a few clusters with more general templates, for example clusters contains many variations of text-only templates. (b) Histogram of total usage per cluster. This histogram shows that many clusters have very low usage values (here these clusters correspond to the isolated templates observed in (a)). Conversely, one cluster exhibits exceptionally high usage due to the grouping of two top-10 templates (one of them is shown for illustration).

4.3 Online Phase Results

In this section, we evaluate the performance of our system during the online phase, specifically the recommendation process using the clusters generated in the offline phase. We begin by defining the baseline methods for comparison, the evaluation metrics, and other relevant test characteristics. Then, we present a series of graphs to demonstrate the effectiveness and quality of our recommendation system.

4.3.1 Experiment Design

Baseline Methods To evaluate the quality and performance of our approach, we compare it against the following baseline methods:

- Most Popular Algorithm (*MostPopular*): This is the simplest possible method, and it is fact what our industrial partner uses currently. This method selects and outputs the top-N most popular templates from $\mathcal{T}^A$. It serves as a basic reference for comparison.
- Classic ClusDiv (*CClusDiv*, [2]): n this method, templates are clustered based on usage data, using the number of clusters equal to the top-N list. Unlike our approach, this method does not have a specific heuristic for choosing which cluster with the minimum CW_i (cluster weight) will increase during each iteration. To account for this, we run the algorithm with five different random seeds to select the cluster with minimum CW_i, producing five distinct versions of the same algorithm for comparison.
- Adapted ClusDiv (*AClusDiv*): This is a modified version of ClusDiv where we retain all the algorithm's properties, but cluster templates using the GMN-based distance measure instead of template usage, and without setting N as the number of clusters. Similar to *CClusDiv*, we use the same five random seeds to handle the selection of clusters with the minimum CW_i during iterations.

Evaluation Metrics For each of the metrics presented below, we evaluate the quality of a recommendation list L^N. To simplify the notation, and since the cluster index is no longer important in the final recommendation list, we denote the recommendation list as $L^N = [t_{i_1}, t_{i_2}, ..., t_{i_N}]$.

Diversity We use the *absolute diversity* metric introduced in [2], defined as:

$$D(L) = \frac{1}{N(N-1)} \sum_{t_i \in L_N} \sum_{t_j \in L_N, t_i \neq t_j} d(t_i, t_j), \tag{2}$$

where $d(t_i, t_j)$ is the distance measure defined in (1). This metric computes the average distance between any pair of templates in L^N, where higher values indicate greater diversity in the list. The range of this metric matches that of $GMN(.)$. We chose to keep our approach straightforward by using this metric instead of the z-diversity measure also introduced in [2], which evaluates the relative diversity of a list compared to the overall diversity of the dataset. This choice was made because we are working with a single dataset where rating values were not used to assess diversity, thereby avoiding the issues associated with $D(L^N)$ highlighted in that study.

Accuracy Metric *Normalized Discounted Cumulative Gain* (*nDCG*) is a well-known ranking accuracy metric for RS. It compares the rankings to an ideal order in which all relevant items are positioned at the top of the list in the corresponding order. In our context, relevance is measured

by the template usage u_i of each template t_i in the recommendation list L^N. Given this, we first define the Discounted Cumulative Gain (DCG):

$$DCG@N(L) = \sum_{k}^{N} u_k / \log_2(k+1), \tag{3}$$

and then we define $nDCG$:

$$nDCG@N = \frac{DCG@N(L)}{DCG@N(L')}, \tag{4}$$

where L' is the recommendation list given by the *MostPopular* recommendation algorithm. With this, we are measuring the overall usage of L^N weighted by a log-function that penalizes u_k when the list is not in decreasing order given by the relevance. Additionally, we make it relative to the best possible list (considering only usage) which is given by *MostPopular* algorithm. Here, the range is $[0, 1]$, where *MostPopular* gives the best possible value of $nDCG = 1$.

Popularity-Bias Metric Analyzing the usage distribution of templates, reordered by decreasing usage (see Fig. 3(d)), we defined a usage threshold of 8000 to distinguish between the short head and the long tail. To assess a method's ability to recommend templates beyond the top 4% of the most frequently used ones, we propose calculating the *long-tail percentage* for a recommendation list L. This metric is defined as the proportion of templates in L with usage below 8000.

Scenarios for Template Distributions Across Clusters As previously mentioned, we observe a short-head long-tail usage distribution. However, to assess the diversification capability of our method more comprehensively, we must also analyze the distribution of the most frequently used templates across clusters.

The cumulative number of clusters utilized to group the top-N templates is illustrated in Fig. 5, which we refer to as the *Original Dataset Distribution* (ODD). In this context, the short head of the distribution (e.g., the top-20 used templates) typically corresponds to a unique cluster for each template. This characteristic is advantageous for any recommendation system, as it facilitates maintaining high diversity by selecting templates with elevated rating values.

To further evaluate our method's diversity performance in a less favorable scenario, we constructed a synthetic distribution where the most frequently used templates are clustered together. This was accomplished by first sorting the clusters in decreasing order based on the number of templates they contain. We then extracted the usage distribution as a list, organized in descending order and excluding template IDs. Finally, we assigned each usage value from this ordered list to a corresponding template within the sorted clusters. This approach ensured that the highest-used templates were grouped together, providing a context for assessing the impact on diversity. We refer to this synthetic dataset as the *Increasing Slope Distribution* (ISD), reflecting its characteristic shape, as depicted in Fig. 5.

In our experiments, we will utilize the $\mathcal{T}$ dataset along with usage values assigned to templates according to these two distinct cluster distributions.

4.3.2 Performance Analysis

In this section, we evaluate the performance of the methods by varying N and CW_{th}. To thoroughly assess the capabilities of each method, both in terms of recommendation quality and computational cost, it is essential to utilize the largest possible set of templates. Therefore, we

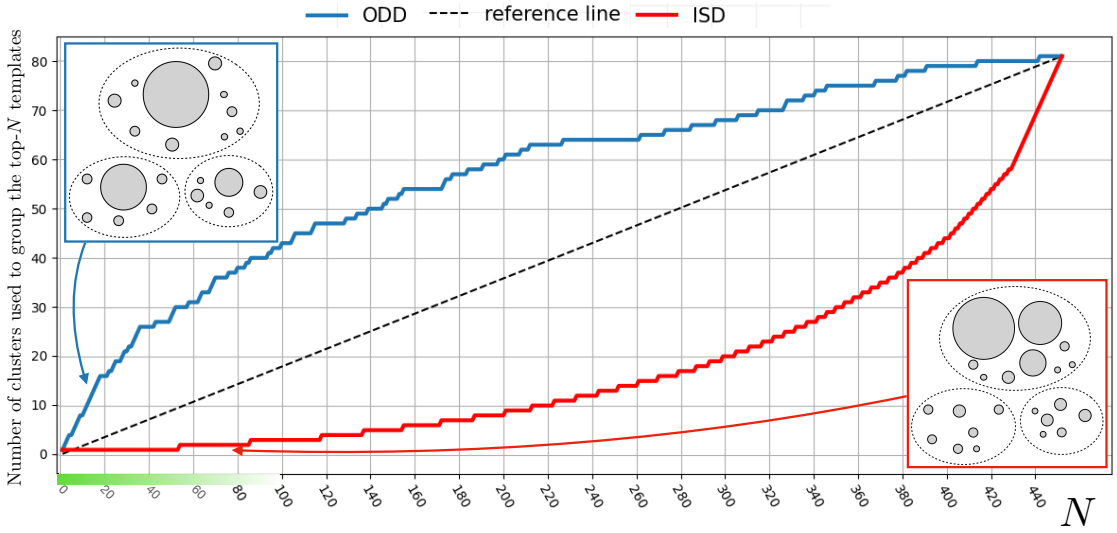

Figure 5: Scenarios for Template Distributions Across Clusters. This figure shows the number of clusters used to group the top-N templates (ranked by usage) for two distinct distribution scenarios. The first scenario, represented by the blue curve, corresponds to the *Original Dataset Distribution* (ODD), where top-N templates are more evenly distributed across clusters, particularly for low N-values (green region). This spread suggests that selecting templates based on usage alone should yield satisfactory diversity. The second scenario, shown by the red curve, simulates a situation where high-usage templates are more frequently concentrated within the same clusters and is termed the *Increasing Slope Distribution* (ISD). This arrangement is intentionally designed to be nearly symmetric relative to the dotted reference line, creating a more challenging scenario, as selecting templates based solely on usage would likely lead to lower diversity in recommendations.

chose not to impose any specific requirements, ensuring that all templates in the dataset are considered admissible.

In Fig. 6, we show results varying CW_{th} with N fixed. Let us comment on the different evaluation metrics.

Diversity: Under the ODD distribution, our method demonstrates slightly better diversity compared to other methods primarily due to the local suppression mechanism.

On the other hand, a notable improvement is observed under the ISD distribution, where the top-N templates are concentrated in common clusters. Here, the CW reassignment method plays a critical role in achieving higher diversity. By reassigning CW_i values based on inter-cluster distances, we substantially enhance the method's diversification power. Additionally, using the local suppression mechanism slightly improves diversity even more. Interestingly, our method increases diversity until it peaks because smaller CW_{th} values necessitate more iterations of the CW reassignment, which incrementally increases diversity locally (i.e., relative to the source cluster). However, global diversity (i.e., the overall distance to all clusters within the top-N list) may be negatively impacted. This behavior suggests that higher CW_{th} values are required to reach a plateau at a certain point. Beyond this point, diversity begins to decline as too many similar templates are suggested. Nonetheless, our method maintains a higher diversity level than other approaches until it converges with them at $CW_{th} = N$.

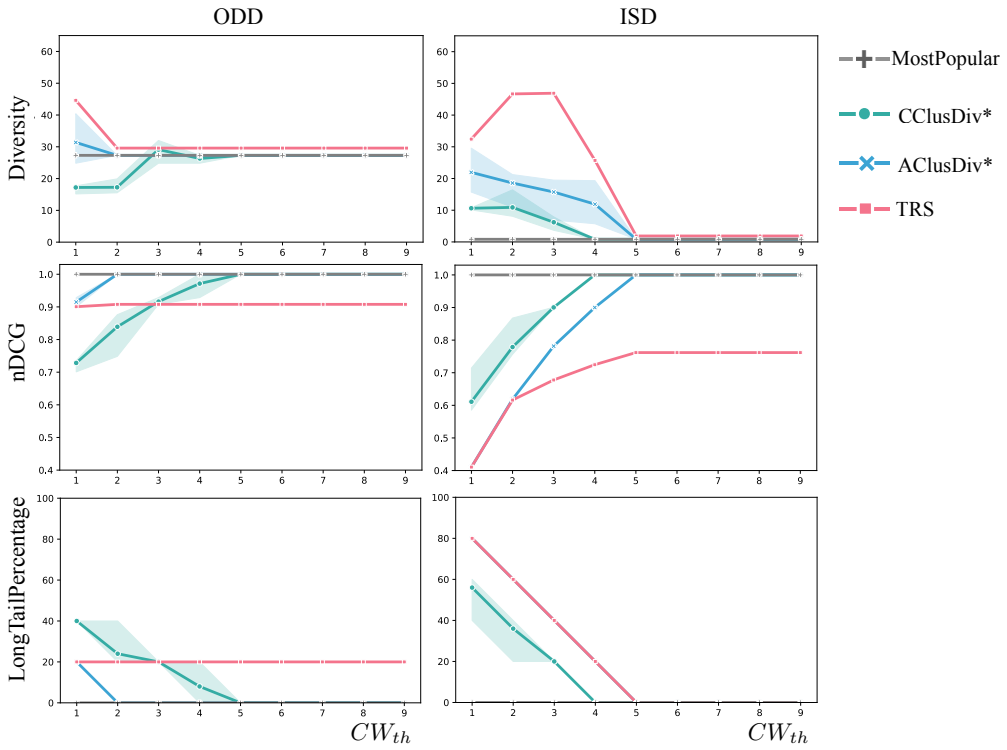

Figure 6: Diversity, $nDCG$ and Long Tail % vs CW_{th} for fixed $N = 5$ Methods with multiple variants ($CClusDiv$ and $AClusDiv$) are represented by their average values (colored dots) and a shaded region indicating the range. These methods are marked with a star (*) in the legend. Note that for $CW_{th} > N$, all algorithms stabilize as no further reassignment occurs.

Accuracy: Enhancing diversity through the selection process impacts accuracy ($nDCG$) in comparison to *CClusDiv* and *AClusDiv*, as observed in both ODD and ISD distributions; this is because the local suppression mitigates the usage of popular templates that are too similar to the ones already selected. This trade-off between accuracy and diversity is well-documented in recommendation systems literature [18]. However, in our application, diversity holds greater importance, and we prioritize this property over achieving higher $nDCG$ values. Nevertheless, our method maintains acceptable accuracy levels, ensuring a balanced performance.

Long Tail Percentage: When selecting templates based solely on usage (MostPopular method), only the top 4% of templates are chosen, completely neglecting the remaining 96%, resulting in zero long-tail usage. For *CClusDiv*, long-tail usage is high when $CW_{th} < 3$ because clustering by usage forces the method to draw from less-popular clusters. However, this effect diminishes as CW_{th} increases, allowing the method to revert to more popular clusters. In contrast, *AClusDiv* struggles to promote long-tail usage since it does not cluster by usage, and the CW reassignment step alone has limited impact. To address this, our method introduces a selection process that actively promotes less-popular templates within clusters. For ODD, this approach consistently includes at least one low-popularity template, regardless of CW_{th}, due to the local suppression mechanism. For ISD, our method achieves higher long-tail usage than others across all CW_{th} values, though this decreases for larger CW_{th}, as fewer constraints force the selection of clusters with low usage. Despite not clustering directly by usage, ISD exhibits similar properties due to the way clusters are constructed.

In Fig. 7, we show results varying N with CW_{th} fixed. Let us comment again on the different evaluation metrics.

Diversity : Our method demonstrates slightly better diversity than the other methods for the ODD distribution, thanks to its selection step. Furthermore, it significantly outperforms the others in the ISD case. As observed when varying CW_{th}, diversity reaches its peak when $N \approx 2CW_{th}$, for the same reasons discussed previously.

Accuracy: In terms of $nDCG$, we observe the same trade-off as before: improving diversity comes at the cost of lower $nDCG$ values compared to baseline methods. Notably, *CClusDiv* experiences a significant drop in quality when $CW_{th} < N$, as it clusters based on usage with the number of clusters set to N. This heavily impacts $nDCG$ during the reassignment step, as the system is forced to select from very low-popularity templates.

Long Tail Percentage: The results align with the previous analysis: our selection process effectively promotes the use of less popular templates within each cluster, outperforming the baseline methods.

Execution Time: In Fig. 8 we analyze execution time (in milliseconds) for varying N, keeping CW_{th} fixed.

For simplicity, we present results for varying N only, as the trends observed when varying CW_{th} are equivalent.

Our method has a slower execution time compared to *AClusDiv*, as expected, due to the computational overhead of calculating distances between clusters during the CW_i reassignment step. However, both our method and *AClusDiv* outperform *CClusDiv* in terms of execution time for ODD and ISD distributions, as neither requires clustering templates during runtime;

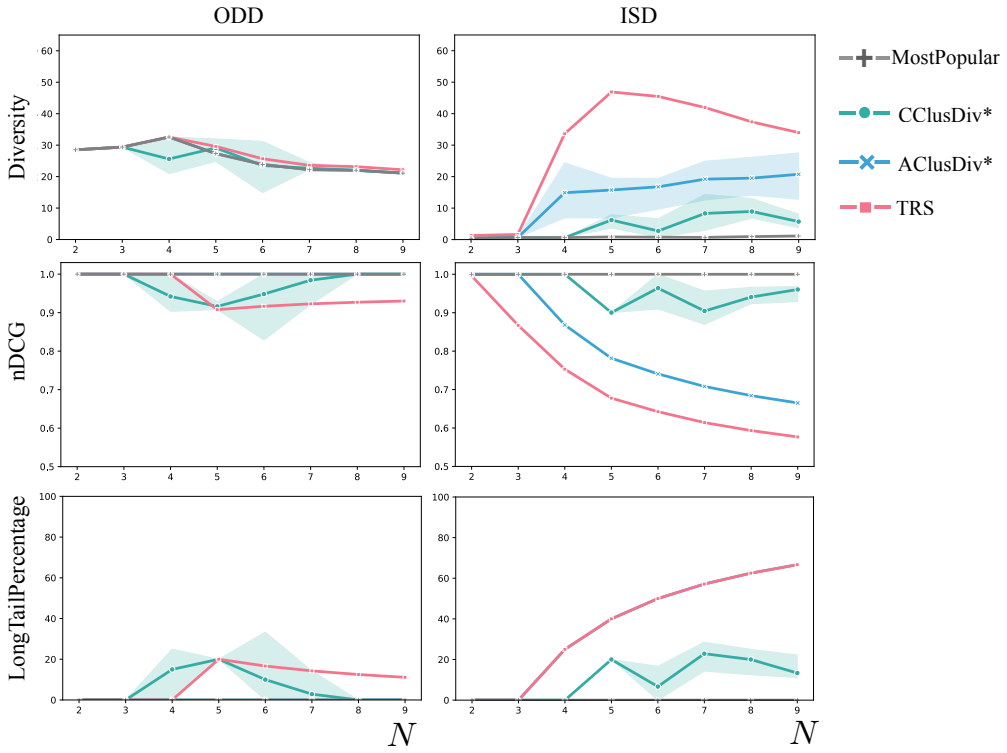

Figure 7: Diversity, $nDCG$ and Long Tail % vs N for fixed $CW_{th} = 3$. The legend and representation of methods with multiple variants ($CClusDiv$ and $AClusDiv$) follow the same convention as in Fig.6, where averages are shown with colored dots and ranges are shaded. Note that for $N > CW_{th}$, all algorithms stabilize as no further reassignment occurs.

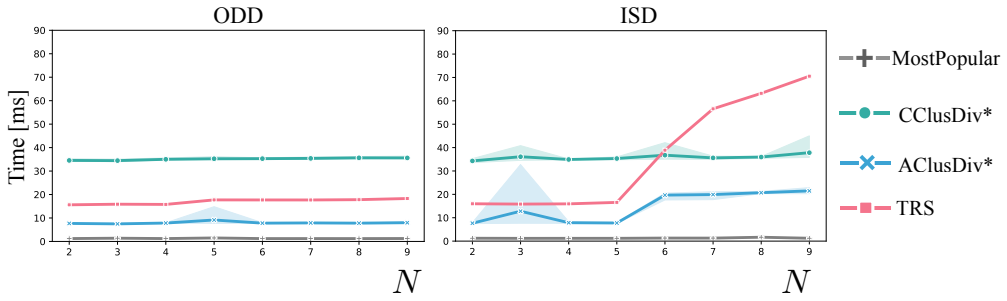

Figure 8: Time vs N for fixed $CW_{th} = 3$. The legend and representation of methods with multiple variants ($CClusDiv$ and $AClusDiv$) are the same as in Fig. 6.

instead, clustering is handled in a preprocessing phase. *CClusDiv*, on the other hand, clusters templates dynamically using N as the number of clusters, increasing its computational cost.

For the ISD dataset, our method's execution time rises with increasing N, especially when $N > CW_{th}$, eventually surpassing *CClusDiv* due to the distance calculations required for CW reassignment. Nonetheless, this additional cost is justified by the significant diversity improvements achieved. Importantly, our method maintains execution times within the same order of magnitude (milliseconds) and avoids runtime clustering, making it advantageous for larger datasets.

4.3.3 Example Outputs

Figures 9 and 10 showcase two query examples that highlight the performance of our recommendation system compared to baseline methods. To reflect real-world conditions, these examples are based on the original dataset distribution (ODD). They demonstrate how our method successfully promotes diversity while aligning with user preferences. In both cases, *CClusDiv* struggles to achieve template diversity due to its lack of structural layout consideration. While *AClusDiv* improves diversity by accounting for layout structure, our approach goes further by leveraging cluster distances and an optimized selection process to maximize structural differences and enhance diversity.

5 Discussion

5.1 Top-N Preservation

A desired property of our method is that Top-N list L^N and Top-$N+1$ recommendation list L^{N+1} are actually equivalent but with an "extra" template. More formally:

Proposition 1 $L^N \subset L^{N+1}$.

The proof of this proposition is provided in the Appendix, Sec.A.3. It is worth noting that this property guarantees that all templates in L^N are included in L^{N+1}, although their ordering may differ.

5.2 Generalization

A key question is whether our recommendation method, which uses a template dataset from a single newspaper publisher to train *GMN*, can generalize to other datasets or template types. While our experiments were conducted on this specific dataset, our approach is inherently designed with adaptability in mind. This is rooted in two key aspects: (1) the *GMN* model is trained to learn a generalizable similarity measure between any pair of templates, not restricted to those in the training set. By identifying layout similarities that transcend specific datasets, *GMN* can theoretically be applied to templates from diverse sources without modification; and (2) our dataset encompasses a wide range of templates, from complex layouts featuring six to eight images to minimalistic, text-only designs. This diversity enables our recommendation model to detect both major and subtle structural differences, equipping it to handle new and varied template sets effectively. Collectively, these factors indicate that our method, while validated on a single publisher's dataset, has the potential to generalize across different template databases and to be extended to other types of structured, geometrically organized data.

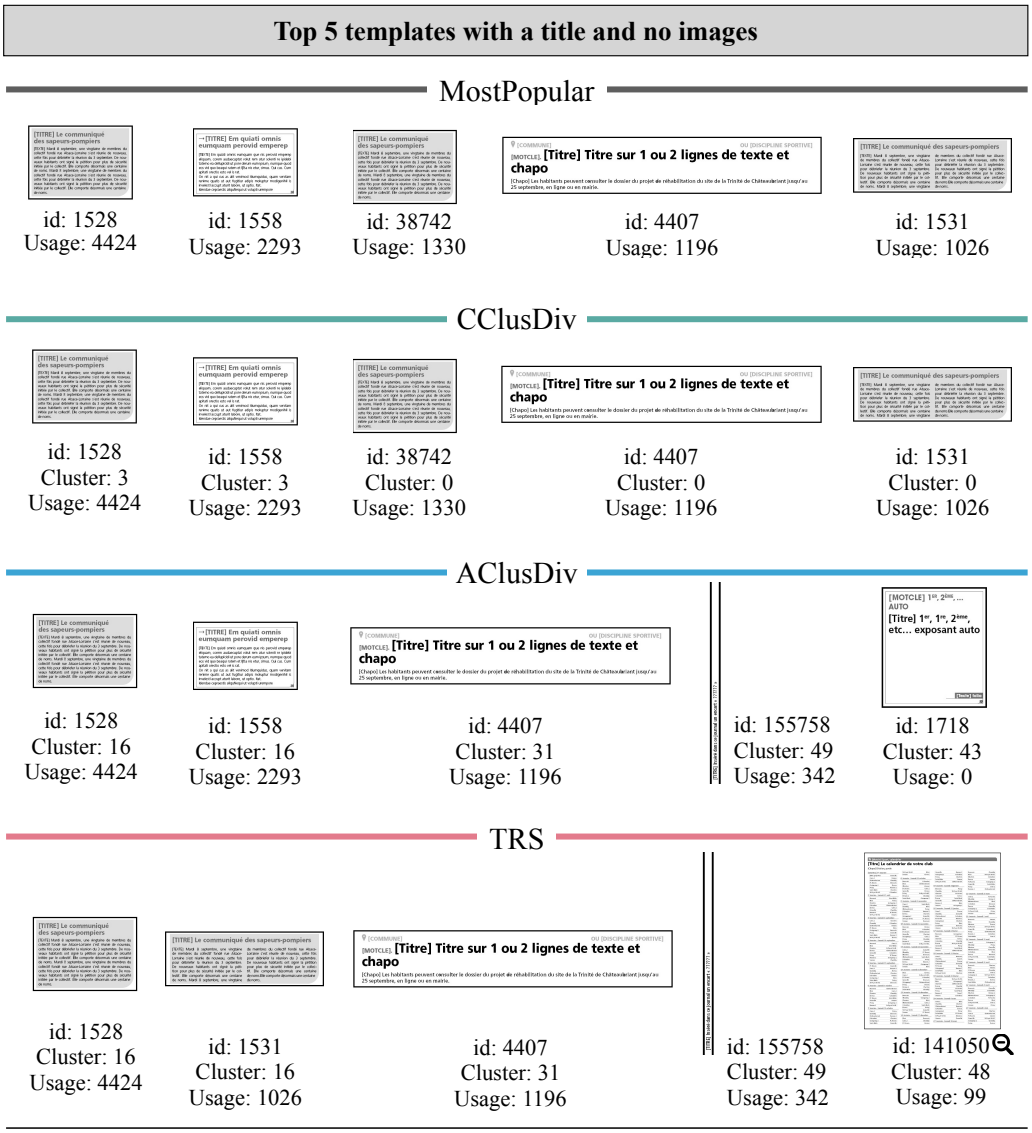

Figure 9: Comparison of the top 5 recommended templates with a title and no images, given by baseline methods versus our proposed method. The figure highlights the differences in diversity and relevance of template selection, demonstrating the effectiveness of our method in meeting the specified query criteria. In this example, the clustering weight threshold parameter is set to $CW^{th} = 2$. For convenience, template number 141050 (last in the figure) is shown at a reduced size to fit the layout.

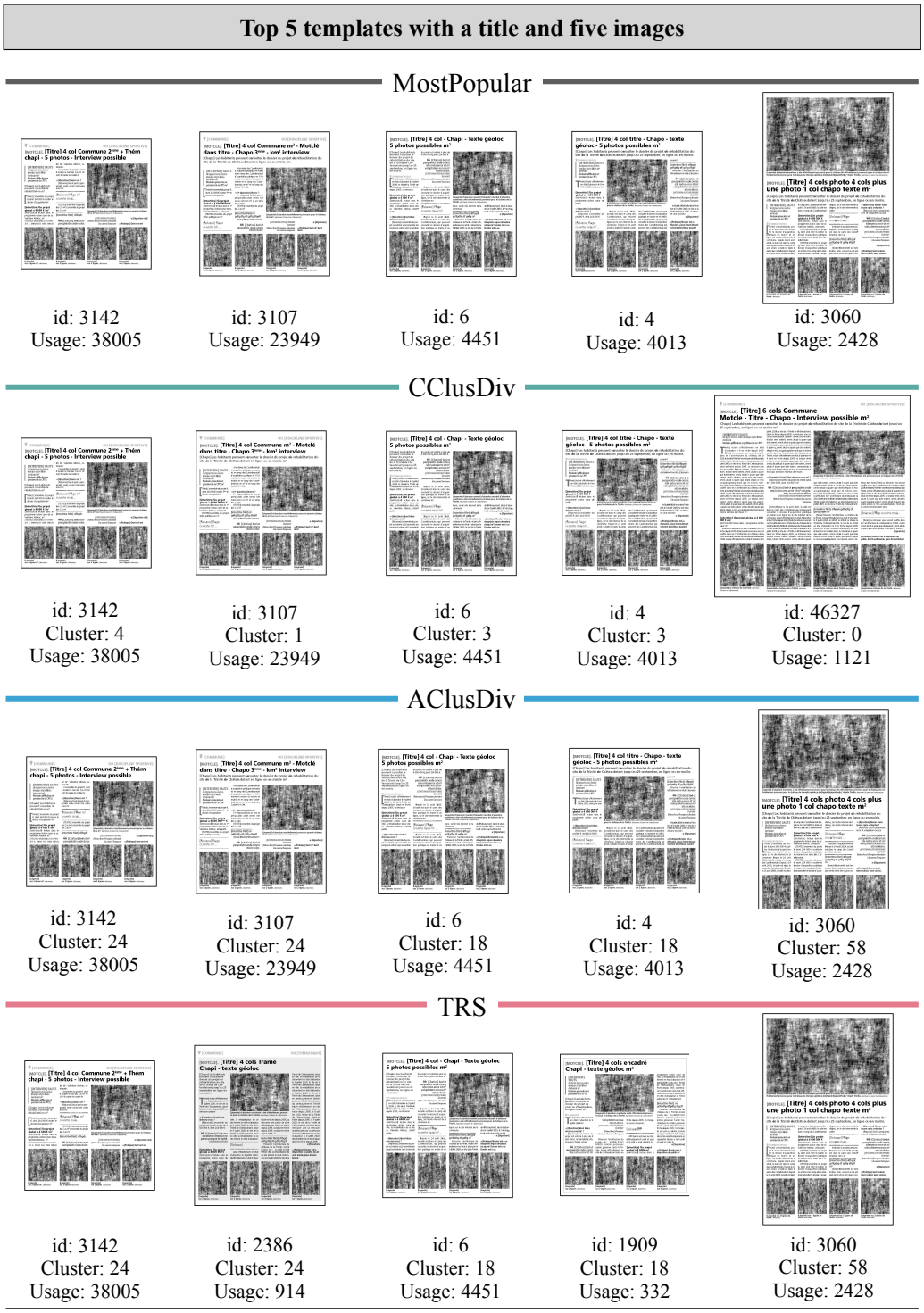

Figure 10: Comparison of the top 5 recommended templates with a title and five images, given by baseline methods and our proposed method, as in Fig. 9. This figure further illustrates the diversity and relevance of template selection obtained with our method. This example uses the same cluster weight threshold ($CW^{th} = 2$).

5.3 User/Item Application

Our system operates in a setting without explicit individual user preferences—often referred to as a user-agnostic scenario. In this context, recommendations are based purely on template attributes, tailored for designers who need templates suitable for specific use cases. However, our method can be generalized to incorporate user-specific preferences in a user-aware setting, where individual usage data is captured in a template-user matrix. In such cases, our method could leverage collaborative filtering (CF) techniques to enhance recommendations [34]. For instance, a user-based CF approach could estimate rating values by identifying similarities in user behaviors. Alternatively, item-based CF methods could be employed, using a distinct similarity measure (such as cosine similarity) for the CF step. This differentiation is necessary because the *GMN* model evaluates structural layout similarity, which does not necessarily correlate with user preferences or template popularity. This adaptability highlights the potential for extending our approach to user-centered scenarios.

5.4 Practical Considerations for Deployment

To ensure practical deployment in real-world scenarios, an essential consideration is the computational complexity of our method. We evaluate this aspect across three primary steps: (i) Offline Phase: Training the *GMN* model is computationally intensive, typically requiring several hours. However, this step is performed only once per dataset and is not executed in real-time. Importantly, retraining is unnecessary when new templates are added, as the pre-trained network is already capable of recognizing structural layout differences. For similar datasets, pre-trained networks can often be reused or fine-tuned, avoiding the need for full retraining (see Sec. 5.2). (ii) Clustering Step: Calculating the distance matrix for templates takes a few minutes, while the agglomerative clustering process requires only 1–2 seconds. This step is performed once per dataset and each time new templates are added, but does not need to be repeated in real-time. In fact, one can schedule an update routine periodically to keep the dataset up to date. (iii) Online Phase: The recommendation process is extremely efficient, with response times in the range of milliseconds, as demonstrated in our experiments. This ensures the method's viability for practical deployment.

6 Conclusion

We presented a clustering-based recommendation system designed to promote diversity in template recommendations by integrating both user preferences and the content of the written article. Our approach includes an offline phase, where clusters are generated using a similarity metric based on LayoutGMN, a graph-matching neural network specifically trained to evaluate structural similarities between templates. In the online phase, templates are recommended by combining predicted ratings or usage data with a diversity-promoting mechanism that selects templates from different clusters, maximizing the distance between them to enhance variety in layout options. Our method demonstrated significant improvements over *CClusDiv* and *AClusDiv*, especially in scenarios where high-rated items are concentrated within a few clusters rather than distributed broadly. Finally, empirical use cases with varying user requirements illustrated the system's qualitative effectiveness in enhancing recommendation diversity and relevance.

Although our work provides an initial solution for optimizing article template selection, it represents just the beginning of a broader exploration of how recommendation systems can transform the press industry. In particular, future work should extend beyond individual article templates to include recommendations for full-page layouts and entire newspapers, considering

the complexities of multi-page design. To fully realize the potential of recommendation systems in this domain, further collaborations with industry stakeholders are essential. These partnerships will be critical for refining the system, improving its relevance, and ensuring its applicability across the various stages of newspaper production, ultimately leading to a more efficient and effective workflow. We would like to thank the publications of the Publihebdos press group for allowing us to use the models of their graphic charter. Their contribution was invaluable and enabled us to work within a real production framework.

Bibliography

[1] Gediminas Adomavicius and YoungOk Kwon. 2012. Improving Aggregate Recommendation Diversity Using Ranking-Based Techniques. *IEEE Transactions on Knowledge and Data Engineering* 24, 5 (2012), 896–911. https://doi.org/10.1109/TKDE.2011.15

[2] Tevfik Aytekin and Mahmut Özge Karakaya. 2014. Clustering-based diversity improvement in top-N recommendation. *Journal of Intelligent Information Systems* 42, 1 (Feb. 2014), 1–18. https://doi.org/10.1007/s10844-013-0252-9

[3] Bing Bai, Yushun Fan, Wei Tan, and Jia Zhang. 2020. DLTSR: A Deep Learning Framework for Recommendations of Long-Tail Web Services. *IEEE Transactions on Services Computing* 13, 1 (2020), 73–85. https://doi.org/10.1109/TSC.2017.2681666

[4] Christine Bauer, Chandni Bagchi, Olusanmi A. Hundogan, and Karin van Es. 2024. Where Are the Values? A Systematic Literature Review on News Recommender Systems. *ACM Trans. Recomm. Syst.* 2, 3, Article 23 (jun 2024), 40 pages. https://doi.org/10.1145/3654805

[5] Irina Beregovskaya and Mikhail Koroteev. 2021. Review of Clustering-Based Recommender Systems. *CoRR* abs/2109.12839 (2021). arXiv:2109.12839 https://arxiv.org/abs/2109.12839

[6] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. 2013. Recommender systems survey. *Knowledge-Based Systems* 46 (2013), 109–132. https://doi.org/10.1016/j.knosys.2013.03.012

[7] Pablo Boczkowski. 2004. *Digitizing the News.*

[8] Yan Chen, Emilian Vankov, Linas Baltrunas, Preston Donovan, Akash Mehta, Benjamin Schroeder, and Matthew Herman. 2023. Contextual Multi-Armed Bandit for Email Layout Recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems* (Singapore, Singapore) *(RecSys '23)*. Association for Computing Machinery, New York, NY, USA, 400–402. https://doi.org/10.1145/3604915.3608878

[9] Luciano da F Costa. 2021. Further generalizations of the Jaccard index. *arXiv preprint arXiv:2110.09619* (2021).

[10] Yashar Deldjoo, Markus Schedl, Paolo Cremonesi, and Gabriella Pasi. 2020. Recommender Systems Leveraging Multimedia Content. *Computing Surveys* 53 (09 2020), 1–38. https://doi.org/10.1145/3407190

[11] Javier Errea. 2018. *Newspaper Design: Editorial Design from the World's Best Newsrooms.* Gestalten.

[12] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, and Yong Li. 2022. A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions. *ACM Transactions on Recommender Systems (TORS)* (2022).

[13] Neil Hurley and Mi Zhang. 2011. Novelty and Diversity in Top-N Recommendation – Analysis and Evaluation. *ACM Trans. Internet Techn.* 10 (03 2011), 14. https://doi.org/10.1145/1944339.1944341

[14] Ali Jahanian, Jerry Liu, Qian Lin, Daniel Tretter, Eamonn O'Brien-Strain, Seungyon Claire Lee, Nic Lyons, and Jan Allebach. 2013. Recommendation system for automatic design of magazine covers. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces* (Santa Monica, California, USA) *(IUI '13)*. Association for Computing Machinery, New York, NY, USA, 95–106. https://doi.org/10.1145/2449396.2449411

[15] Wensen Jiang, Yizhu Jiao, Qingqin Wang, Chuanming Liang, Lijie Guo, Yao Zhang, Zhijun Sun, Yun Xiong, and Yangyong Zhu. 2022. Triangle Graph Interest Network for Click-through Rate Prediction. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining* (Virtual Event, AZ, USA) *(WSDM '22)*. Association for Computing Machinery, New York, NY, USA, 401–409. https://doi.org/10.1145/3488560.3498458

[16] Joseph Johnson and Yiu-Kai Ng. 2017. Enhancing long tail item recommendations using tripartite graphs and Markov process. In *Proceedings of the International Conference on Web Intelligence* (Leipzig, Germany) *(WI '17)*. Association for Computing Machinery, New York, NY, USA, 761–768. https://doi.org/10.1145/3106426.3106439

[17] H. W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly* 2, 1-2 (1955), 83–97. https://doi.org/10.1002/nav.3800020109 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.3800020109

[18] Matevž Kunaver and Tomaž Požrl. 2017. Diversity in recommender systems – A survey. *Knowledge-Based Systems* 123 (May 2017), 154–162. https://doi.org/10.1016/j.knosys.2017.02.009

[19] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. 2019. Graph Matching Networks for Learning the Similarity of Graph Structured Objects. http://arxiv.org/abs/1904.12787 arXiv:1904.12787 [cs, stat].

[20] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Fi-GNN: Modeling Feature Interactions via Graph Neural Networks for CTR Prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (Beijing, China) *(CIKM '19)*. Association for Computing Machinery, New York, NY, USA, 539–548. https://doi.org/10.1145/3357384.3357951

[21] Han Liu, Yinwei Wei, Jianhua Yin, and Liqiang Nie. 2023. HS-GCN: Hamming Spatial Graph Convolutional Networks for Recommendation. *IEEE Transactions on Knowledge and Data Engineering* 35, 6 (2023), 5977–5990. https://doi.org/10.1109/TKDE.2022.3158317

[22] Siyi Liu and Yujia Zheng. 2020. Long-tail Session-based Recommendation. In *Proceedings of the 14th ACM Conference on Recommender Systems* (Virtual Event, Brazil) *(RecSys '20)*. Association for Computing Machinery, New York, NY, USA, 509–514. https://doi.org/10.1145/3383313.3412222

[23] Weiwen Liu, Qing Liu, Ruiming Tang, Junyang Chen, Xiuqiang He, and Pheng Ann Heng. 2020. Personalized Re-ranking with Item Relationships for E-commerce. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (Virtual Event, Ireland) *(CIKM '20)*. Association for Computing Machinery, New York, NY, USA, 925–934. https://doi.org/10.1145/3340531.3412332

[24] Andrew Luke, Joseph Johnson, and Yiu-Kai Ng. 2018. Recommending Long-Tail Items Using Extended Tripartite Graphs. In *2018 IEEE International Conference on Big Knowledge (ICBK)*. 123–130. https://doi.org/10.1109/ICBK.2018.00024

[25] Nick Newman. 2024. *Journalism, media, and technology trends and predictions 2024*. https://reutersinstitute.politics.ox.ac.uk/journalism-media-and-technology-trends-and-predictions-2024#header--0

[26] Yoon-Joo Park. 2013. The Adaptive Clustering Method for the Long Tail Problem of Recommender Systems. *IEEE Transactions on Knowledge and Data Engineering* 25, 8 (2013), 1904–1915. https://doi.org/10.1109/TKDE.2012.119

[27] Yoon-Joo Park and Alexander Tuzhilin. 2008. The long tail of recommender systems and how to leverage it. In *Proceedings of the 2008 ACM Conference on Recommender Systems* (Lausanne, Switzerland) *(RecSys '08)*. Association for Computing Machinery, New York, NY, USA, 11–18. https://doi.org/10.1145/1454008.1454012

[28] Akshay Gadi Patil, Manyi Li, Matthew Fisher, Manolis Savva, and Hao Zhang. 2021. LayoutGMN: Neural Graph Matching for Structural Layout Similarity. http://arxiv.org/abs/2012.06547 arXiv:2012.06547 [cs].

[29] Jing Qin and Danfeng Hong. 2021. A Survey of Long-Tail Item Recommendation Methods. *Wirel. Commun. Mob. Comput.* 2021 (jan 2021), 14 pages. https://doi.org/10.1155/2021/7536316

[30] Shaina Raza and Chen Ding. 2020. News recommender system: a review of recent progress, challenges, and opportunities. *Artificial Intelligence Review* 55 (2020), 749 – 800. https://api.semanticscholar.org/CorpusID:235790388

[31] Marco Tulio Ribeiro, Nivio Ziviani, Edleno Silva De Moura, Itamar Hata, Anisio Lacerda, and Adriano Veloso. 2015. Multiobjective Pareto-Efficient Approaches for Recommender Systems. *ACM Trans. Intell. Syst. Technol.* 5, 4, Article 53 (dec 2015), 20 pages. https://doi.org/10.1145/2629350

[32] Rama Syamala Sreepada and Bidyut Kr. Patra. 2020. Mitigating long tail effect in recommendations using few shot learning technique. *Expert Syst. Appl.* 140, C (Feb. 2020), 17 pages. https://doi.org/10.1016/j.eswa.2019.112887

[33] Rama Syamala Sreepada and Bidyut Kr. Patra. 2021. Enhancing long tail item recommendation in collaborative filtering: An econophysics-inspired approach. *Electronic Commerce Research and Applications* 49 (2021), 101089. https://doi.org/10.1016/j.elerap.2021.101089

[34] Xiaoyuan Su and Taghi M. Khoshgoftaar. 2009. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence* 2009, 1 (2009), 421425. https://doi.org/10.1155/2009/421425 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1155/2009/421425

[35] Yixin Su, Rui Zhang, Sarah M. Erfani, and Junhao Gan. 2021. Neural Graph Matching based Collaborative Filtering. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 849–858. https://doi.org/10.1145/3404835.3462833

[36] The CGAL Project. 2024. *CGAL User and Reference Manual* (5.6.1 ed.). CGAL Editorial Board. https://doc.cgal.org/5.6.1/Manual/packages.html

[37] Sanne Vrijenhoek, Savvina Daniil, Jorden Sandel, and Laura Hollink. 2024. Diversity of What? On the Different Conceptualizations of Diversity in Recommender Systems. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency* (Rio de Janeiro, Brazil) *(FAccT '24)*. Association for Computing Machinery, New York, NY, USA, 573–584. https://doi.org/10.1145/3630106.3658926

[38] Amy Schmitz Weiss and Carla Schwingel. 2008. THE DELICATE RELATIONSHIP IN JOURNALISM: Where content and production meet in the content management system. A comparative study of US and Brazil Newsroom Operations. *Brazilian journalism research* 4, 2 (2008), 88–107.

[39] Hongzhi Yin, Bin Cui, Jing Li, Junjie Yao, and Chen Chen. 2012. Challenging the long tail recommendation. *Proc. VLDB Endow.* 5, 9 (May 2012), 896–907. https://doi.org/10.14778/2311906.2311916

[40] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 974–983. https://doi.org/10.1145/3219819.3219890

[41] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. 2019. Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. http://arxiv.org/abs/1911.08287 arXiv:1911.08287 [cs].

A Appendix

A.1 Classes for Each Element in a Template

In our system, the elements of each template are assigned to the following classes: *TITRE* (for titles), *SOUSTITRE* (for subtitles), *PHOTO* (for images), *LEGENDE* (for captions), *CHAPO* (for lead phrases), *EXERGUE* (for quotes), *THEMATIQUE/COMMUNE* (reserved for tags indicating city, town and category of the article), *TEXTE* (for body text), and other minor categories grouped as OTHERS.

A.2 Definition of the Similarity Measure *tdIoU*

In this section, we present the general formula for calculating the template-based Intersection over Union (*tdIoU*), a similarity measure used to assess the similarity between two templates based on the layout of their elements, that was used to weakly label pairs and triplets for *GMN* training. The *tdIoU* is calculated by summing a similarity measure *dIoU'* (see Sec. A.2.1) between matched elements of the same class, after rescaling the templates to ensure fair comparison across different dimensions. Mathematically, given two templates A and B, *tdIoU* is defined by:

$$tdIoU(A,B) = C_S\ C_M\ \frac{\sum_{(a_i,\tilde{b}_j)\in M} dIoU'(a_i,\tilde{b}_j)}{|M|}, \tag{5}$$

where the sum is done other all matched pairs of elements denoted by M (where $a_i \in A$ has been matched to $\tilde{b}_j \in \tilde{B}$), where $\tilde{B}$ is a rescaled version of B to have the same dimensions as A (i.e., $H_{\tilde{B}} = H_A$ and $W_{\tilde{B}} = W_A$), C_S is a penalty factor related to the rescaling (see Sec. A.2.2), and C_M is a penalty factor related to the matching between elements (see Sec. A.2.3). In the following paragraphs, we detail the individual steps of this process, including the definition of the *dIoU'* measure, the rescaling procedure, and the element matching algorithm.

A.2.1 Definition of the Similarity Measure *dIoU'*

To ensure compatibility with our penalization factors, we require positive values for the similarity measure. Here we start from *dIoU* introduced in [41] which ranges from $[-1, 1]$. To address this, we shift it by $+1$, resulting in a modified measure $dIoU'(a_i, b_j) = dIoU(a_i, b_j) + 1$. This simple transformation ensures that the similarity measure remains non-negative, preventing potential issues when applying penalties. While not a central contribution of our work, it is worth noting that we implemented *dIoU* efficiently using the Computational Geometry Algorithms Library (CGAL) [36] to handle boolean operations between any pair of polygons representing.

A.2.2 Rescaling and Penalty for Dimensional Differences

Aligning the dimensions facilitates meaningful element-by-element comparisons between templates with different dimensions. So the first step in our method is to rescale template B to match dimensions of template A. We denote this rescaled template $\tilde{B}$ (see example in Fig. 11). Note that resizing A or B is mathematically equivalent. However, the extent of this resizing

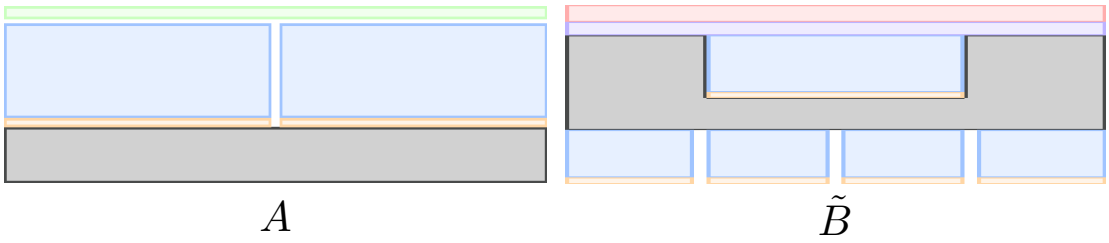

Figure 11: Illustration of rescaling. The template B from Fig. 1 is rescaled to match A-dimensions.

must be accounted for in the similarity calculation. Intuitively, the more the dimensions of B need to be adjusted to align with A, the less similar the two templates should be considered. To capture this effect, we introduce a penalty factor C_S, which reduces the similarity score when there are significant differences in geometry, such as area, aspect ratio, or orientation, between A and B. The penalty term C_S is defined as follows:

$$\begin{aligned} C_S = & f_{\sigma_1,\gamma_1}(H_A W_A - H_B W_B) \times \\ & f_{\sigma_2,\gamma_2}(\arctan(H_A/W_A) - \arctan(H_B/W_B)) \times \\ & g_{\sigma_3,\gamma_3}((\arctan(H_A/W_A) - \pi/4)(\arctan(H_B/W_B) - \pi/4)), \end{aligned} \tag{6}$$

with

$$f_{\sigma,\gamma}(t) = \exp\left(\frac{-(t/\gamma)^2}{2\sigma^2}\right), \quad \text{and} \quad g_{\sigma,\gamma}(t) = \begin{cases} \exp\left(\frac{t/\gamma}{\sigma}\right), & \text{if } t < 0, \\ 1, & \text{if } t \geq 0. \end{cases} \tag{7}$$

This term C_S ensures smooth penalization based on the geometric differences. The first term in C_S penalizes changes in area, the second penalizes deviations in aspect ratio, and the third accounts for the orientation differences between the templates. Together, these factors ensure that templates requiring significant resizing are appropriately penalized in the similarity computation.

A.2.3 Matching Elements

This section explains how elements are matched between templates A and $\tilde{B}$ to compute the summation in (5). The objective is to pair each element in one template with a corresponding element in the other, ensuring that no pair is repeated. The pairing is based on proximity, which is measured using the *dIoU'* similarity metric introduced earlier, and if they correspond to the same class. To achieve this, we formulate the task as an instance of the *assignment problem*, which can be solved efficiently in polynomial time using the Hungarian algorithm [17]. The resulting set of matched pairs is denoted as M. However, in practice, templates A and B often differ in the number and types of elements, so a perfect one-to-one matching is usually impossible. To account for these discrepancies, we introduce a penalty factor, C_M, which quantifies the extent of unmatched elements as follows:

$$C_M = \frac{1}{2area(A)} \sum_{(a_i,\tilde{b}_j)\in M} area(a_i) + area(\tilde{b}_j). \tag{8}$$

This matching process, combined with the penalty factor C_M, ensures that the similarity computation accurately reflects both the quality of matched elements and the structural differences arising from unmatched ones, providing a robust basis for the *tdIoU* measure.

A.3 Proof of Proposition 1

Given a recommendation list L^N let us denote by t_i^N the template in the ith position and its corresponding usage as u_i^N. The proof of Proposition 1 will rely on the two following lemmas.

Let us assume that we have two cluster weight distributions $\{CW_j^N\}$ and $\{CW_j^{N+1}\}$ used to obtain two recommendation lists L^N and L^{N+1} applying the selection step defined Algorithm 1 (lines 19–32). If there exists a unique k such that they satisfy the property $(\mathcal{P}_k)$ defined by:

$$(\mathcal{P}_k) \quad \begin{cases} CW_k^{N+1} = CW_k^N + 1, \\ CW_j^{N+1} = CW_j^N, \quad \forall j \neq k, \end{cases} \tag{9}$$

then we have:

$$L^N \subset L^{N+1}. \tag{10}$$

Let us consider one iteration of the selection step to obtain the ith element. For L^N, we choose template t_i^N from cluster $\mathcal{C}_p$. Similarly, for L^{N+1}, we choose template t_i^{N+1} from cluster $\mathcal{C}_q$.

As long as $p \neq k$ or if $p = k$ but $CW_k^N > 0$, we have that $q = p$ and $t_i^N = t_i^{N+1}$. This can be easily shown by remarking that there is no difference between both executions since there are enough templates in C_p (i.e., the cluster weights are greater than zero for both L^N and L^{N+1} template selection).

Then at some point, we will be in the situation where $p = k$ and $CW_k^N = 0$. In that case, a template t_{i*}^{N+1} will be included in the list L^{N+1} at the $i*$-position (and nothing in the list L^N). Note that this selection may also be accompanied by the suppression of some neighboring

templates in $\mathcal{C}_p$ that were not inhibited in L^N execution. However, it does not matter since t_{i*}^{N+1} is necessarily the last template to be selected from $\mathcal{C}_p$.

After this, in the following iterations, we have $t_i^N = t_{i+1}^{N+1}$ (for all $i \geq i*$), where all templates come from clusters $\mathcal{C}_l$ where $l \neq k$.

So in the end, there is some offset between the two lists due to the inclusion in L^{N+1} of the template t_{i*}^{N+1} at a certain position, which proves (10).

Let us assume that we have two cluster weight distributions $\{CW_j^N\}$ and $\{CW_j^{N+1}\}$. If there exists a unique k_1 such that they satisfy property $(\mathcal{P}_k)$ then, after one iteration of the for loop of reassignment as defined in Algorithm 1 (lines 10–18), there exists a unique k_2 (eventually different from k_1) such that the new cluster weight distributions satisfy property $(\mathcal{P}_k)$.

Considering one iteration of the for loop means that one focuses on a specific source cluster $\mathcal{C}_k$. Three cases can occur.

Case 1: $k_1 = k$, i.e., $\mathcal{C}_{k_1}$ is a the source cluster In that case, there are three possibilities:

(i) If $CW_{k_1}^N < CW_{k_1}^{N+1} < CW_{th}$, then cluster weights for that cluster k are not changed, so that the property $(\mathcal{P}_k)$ is satisfied with $k_2 = k_1$.

(ii) If $CW_{k_1}^{N+1} > CW_{k_1}^N \geq CW_{th}$, then for both executions the reassignment will be the same, i.e., decrease CW_{k_1} and increase CW_j of another cluster $\mathcal{C}_j$ in both cases (for N and $N+1$). Thus at the end of the iteration, we have $CW_j^{N+1} = CW_j^N + 1$ so that the property $(\mathcal{P}_k)$ is satisfied with $k_2 = j$.

(iii) If $CW_{k_1}^N = CW_{th}$, there is no reassignment for L^N, but there is one for L^{N+1} to a target cluster $\mathcal{C}_j$. Thus at the end of the iteration, we have that $CW_{k_1}^{N+1} = CW_{k_1}^N$ and $CW_j^{N+1} = CW_j^N + 1$ so that the property $(\mathcal{P}_k)$ is satisfied with $k_2 = j$.

Case 2: $k_1 \neq k$ but $\mathcal{C}_{k_1}$ is a target cluster In that case, reassignment will increase CW_{k_1} and reduce CW_j of another cluster $\mathcal{C}_j$). Here we have two possibilities.

(i) If $\mathcal{C}_{k_1}$ is a target cluster for both executions (for L^N and L^{N+1}), then $CW_{k_1}^N$ and $CW_{k_1^{N+1}}$ will be increased by one, so we still have $CW_{k_1}^{N+1} = CW_{k_1}^N + 1$. Then the property $(\mathcal{P}_k)$ is satisfied with $k_2 = k_1$.

(ii) If $\mathcal{C}_{k_1}$ is a target cluster for L^N execution and is not for L^{N+1} (which is the only other case possible since $CW_{k_1}^{N+1} > CW_{k_1}^N$ so that it is not possible that $\mathcal{C}_{k_1}$ be a target for L^{N+1} and not for L^N), then only $CW_{k_1}^N$ will be increased by one, and $CW_{k_1}^{N+1} = CW_{k_1}^N$. However, that means another cluster $\mathcal{C}_j$ will be target in L^{N+1} execution, which means that $CW_j^{N+1} = CW_j^N + 1$. So the property $(\mathcal{P}_k)$ is satisfied with $k_2 = j$.

Case 3: Cluster $\mathcal{C}_{k_1}$ is neither a source nor a target cluster Since in this case there is no change for $CW_{k_1}^n$ (for $n = N$ and $n = N + 1$) and that the potential reassignments will be the same for both L^N and L^{N+1}, then the property $(\mathcal{P}_k)$ is satisfied with $k_2 = k_1$.

(Proposition 1) After the initialization step of a set of $\{CW_j\}_{j=1,\ldots,M}$ (M is total number of clusters), is it clear that to form L^{N+1} will require one more template than L^N, increasing only one of the CW_j values. So at initialization, there exists a unique k_0 such that $(\mathcal{P}_k)$ is satisfied.

Then, the reassignment step will change each cluster weight to ensure the diversity of the final recommendation. Thanks to Lemma A.3, we know that after successive iterations i, there will always be a unique k_i such that $(\mathcal{P}_k)$ is satisfied.

Given this, and thanks to Lemma A.3, we can ensure that $L^N \subset L^{N+1}$.

RESEARCH CENTRE
Centre Inria d'Université Côte d'Azur

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399