Leveraging User-Generated Reviews for Recommender Systems with Dynamic Headers

Shanu Vashishtha[a,*,1]**, Abhay Kumar**[a,1]**, Lalitesh Morishetti**[a]**, Kaushiki Nag**[a] **and Kannan Achan**[a]

[a]Personalization and Recommendation, Walmart Global Tech, Sunnyvale, USA

Abstract. E-commerce platforms have a vast catalog of items to cater to their customers' shopping interests. Most of these platforms assist their customers in the shopping process by offering optimized recommendation carousels, designed to help customers quickly locate their desired items. Many models have been proposed in academic literature to generate and enhance the ranking and recall set of items in these carousels. Conventionally, the accompanying carousel title text (header) of these carousels remains static. In most instances, a generic text such as "Items similar to your current viewing" is utilized. Fixed variations such as the inclusion of specific attributes "Other items from a similar seller" or "Items from a similar brand" in addition to "frequently bought together" or "considered together" are observed as well. This work proposes a novel approach to customize the header generation process of these carousels. Our work leverages user-generated reviews that lay focus on specific attributes (aspects) of an item that were favorably perceived by users during their interaction with the given item. We extract these aspects from reviews and train a graph neural network-based model under the framework of a conditional ranking task. We refer to our innovative methodology as Dynamic Text Snippets (DTS) which generates multiple header texts for an anchor item and its recall set. Our approach demonstrates the potential of utilizing user-generated reviews and presents a unique paradigm for exploring increasingly context-aware recommendation systems.

1 Introduction

A recommender system, in the era of e-commerce, serves as a critical tool that offers suggestions personalized to users' tastes and preferences. These systems employ machine learning algorithms to analyze users' behavior and preferences, thereby providing accurate recommendations[23]. The immense value of recommender systems in improving customer experience, enhancing user engagement, and boosting businesses' profitability is undeniable [3, 14]. However, a specific yet significant challenge that arises in the context of these systems is generating dynamic header text for the accompanying carousel in which recommendations are shown to the users. This task involves crafting unique and appealing headers that not only capture the essence of the recommended content but also resonate with individual user preferences.

Generating dynamic header text for a recommendation carousel presents significant challenges that requires a nuanced understanding of both the content and user behaviour. Firstly, capturing user

preferences towards an item is a complex task due to the individual variability and subjectivity involved in preferences, which are often influenced by cognitive, emotional and situational factors. Secondly, accurately providing immediate context and ensuring a direct correlation between the headers and carousel items is necessary for an improved navigation, user engagement, and overall coherence of the digital interface. Lastly, these dynamic headers must positively appeal to users as they serve as the initial point of interaction, influencing user engagement with the carousel items and can significantly impact user satisfaction and retention rates. If these headers are not appealing, user attention may drift leading to decreased interaction, even if the recommendation items are relevant. All these reasons necessitate the development of sophisticated algorithms and robust underlying data analysis techniques demanding a scalable solution.

In this paper, we explore the under examined but promising impact of dynamic header texts. This unique approach can potentially optimize the functionality of recommender systems by offering more tailored and relevant user experiences. A crucial aspect to consider is the provision of straightforward explanations to users. When users understand why certain content is shown to them, their confidence increases, thereby potentially boosting their overall engagement and satisfaction. It offers a novel yet subtle way to potentially improve key business metrics. The benefits of addressing this problem, therefore, are multifaceted and provide a fresh perspective on user interface personalization.

There is a scarcity of literature specifically focused on dynamic header text in recommender systems. However, we came across some related studies in the broader field of personalized user experience. For instance, McNee et al. [8] discusses the importance of explaining recommendations to users. Another research [5] also discusses the potential benefits of personalization and adaptivity in recommender systems. But most of these were focused only on the items shown to the users while none of these studies explicitly address the creation of dynamic header text for recommendations, thus marking an unexplored avenue in the research landscape. Our work aims to bridge this gap by investigating the experiential aspect of dynamic header text. We believe that understanding how this feature can enhance users' confidence and satisfaction levels could provide valuable insights for achieving a personalized header text offering explainable recommendations to users.

In this work, we propose a novel aspect-based recommender system - Dynamic Text Snippets (DTS) that harnesses the power of user-generated reviews to identify and link items through shared aspects. Our approach involves extracting aspect information for items from user reviews and incorporating it into the model training phase, al-

* Corresponding Author. Email: Shanu.Vashishtha@walmart.com
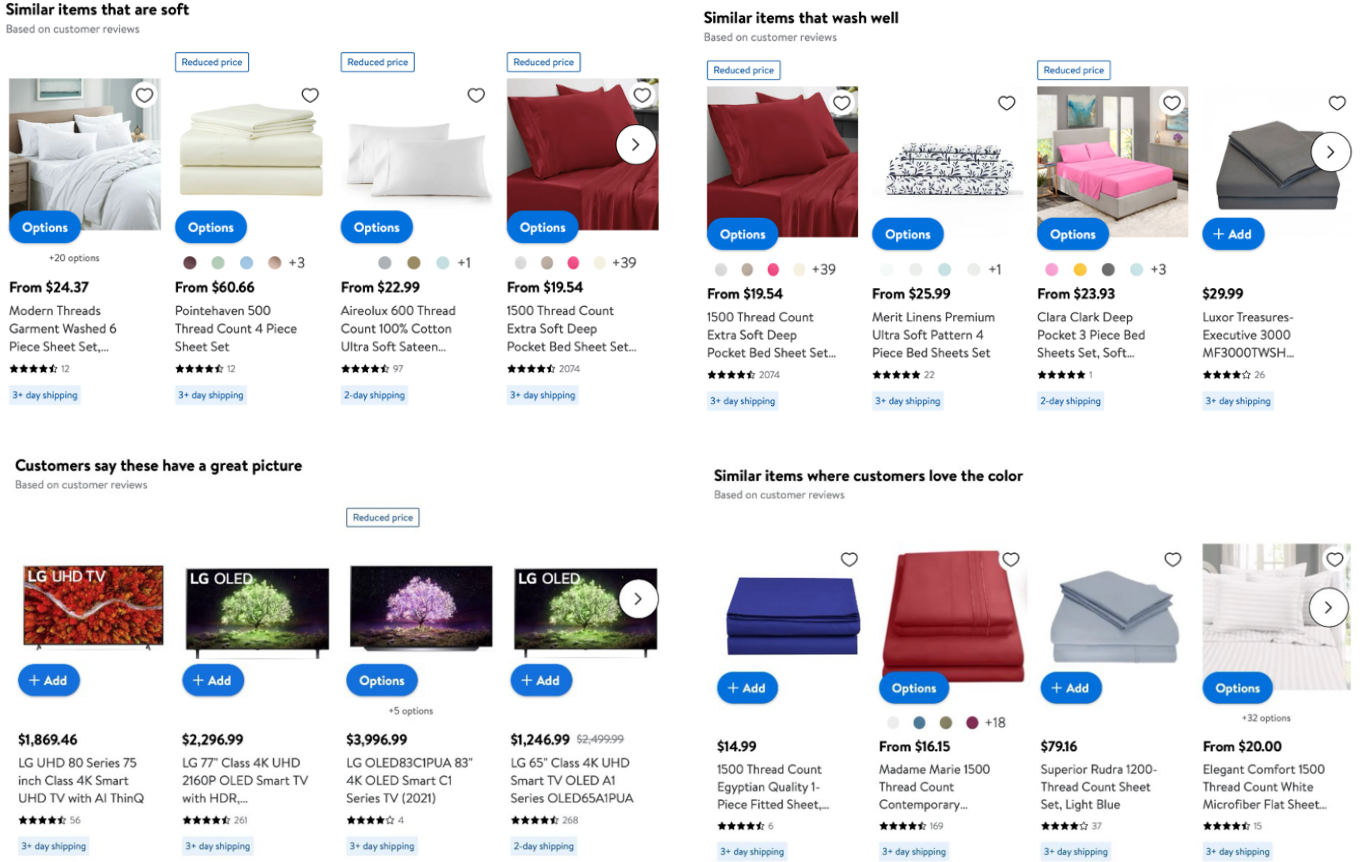[1] Equal contribution.

Figure 1. Examples of Dynamic Text Snippets (DTS) carousel for e-commerce recommendations. Each carousel has recommended items with shared aspects and the associated explainable header.

lowing for a more contextualized and nuanced understanding of user preferences. The aspect extraction includes a preprocessing stage followed by candidate aspect identification. Next, we train a graph neural network-based recommendation model that leverages this aspect information to generate relevant items with multiple header texts for a given set of reference items (anchor items). To evaluate the effectiveness of our proposed DTS model, we conduct a series of offline and online experiments. Our results demonstrate that the incorporation of aspect information leads to significant improvements in the diversity of recommendations.

2 Background and Related Work

2.1 Graph Neural Networks

E-commerce recommender systems have been an area of significant interest over the past few years, owing to their potential to drive customer engagement and sales [3]. Advances in machine learning and the consideration of contextual information have further improved the quality and personalization of recommendations. Zhang et al. [23] proposed a deep learning-based recommender system that combines user-item interactions and user/item metadata for recommendation, demonstrating promising results. Recently, graph neural network (GNN) techniques have been widely utilized in recommender systems [20]. This trend stems from the inherent graph-structured na-

ture of recommendation system data and the effectiveness of GNNs in learning informative representations from graphs.

The GNNs learn expressive representations from graph-structured data by effectively capturing high-order connectivity, going beyond direct user-item interactions to uncover complex patterns. [20] They also exploit the structural properties of the recommendation graph to learn richer representations and offer the flexibility to seamlessly integrate side information (such as item attributes or social relationships) to further refine recommendations. The inherent graph structure of recommender systems, where users and items form nodes with interactions as edges (capturing interactions like clicks, add-to-carts, and transactions), aligns naturally with the capabilities of GNNs.

Graph Attention Networks (GATs) [16] are a significant advancement in graph neural networks, introducing attention mechanisms that enable the model to dynamically learn the importance of different neighbors within a graph structure. This is achieved by assigning learnable attention coefficients to each edge, determining how much a neighbor's features should influence a node's updated representation. GATs offer advantages like adaptive focusing on the most relevant parts of the graph, leading to improved representations. Additionally, they automate the process of determining neighbor importance, reducing the need for manual feature engineering.

Building on the foundation of GATs, Cen et al. [1] addresses the challenge of learning meaningful node embeddings within complex

network structures which are common in real-world applications like e-commerce. Specifically, it focuses on Attributed Multiplex Heterogeneous Networks (AMHENs) where nodes possess diverse types, attributes, and participate in multiple layers based on different relationship types. The authors introduce GATNE, a framework for both transductive (GATNE-T) and inductive (GATNE-I) embedding learning. GATNE integrates node attributes and leverages multi-view relationships characteristic of AMHENs. In this paradigm, nodes are mapped to low-dimensional vectors while preserving network properties. The GATNE model splits each node's embedding into a base embedding shared across edge types, and edge-specific embeddings aggregated from neighbors. GATNE-T is the transductive model using random walks and heterogeneous skip-gram on the network. GATNE-I is the inductive model using node attributes to generate base and edge embeddings via transformation functions.

2.2 Knowledge Graph Embeddings

The use of Knowledge Graphs (KG) [18] has been gaining traction in representing diverse types of information through various entities and relations. The KG is a directed heterogeneous multigraph where node and relation types provide domain-specific semantics. This structure allows for encoding knowledge in a manner interpretable by humans and conducive to automated analysis. In the context of KGs, the conventional graph terminology is replaced with terms like entities for vertices and triplets for directed edges. The triplet is represented as a (h, r, t) tuple in which 'h' is the head entity, 't' is the tail entity, and 'r' is the relation linking them. The term 'relation' here specifies the type of relation, such as 'wants-to-buy' or 'has-bought'. Previous work has shown that a populated KG effectively encodes the knowledge about a particular domain based on the types of nodes and relations included. [12]

2.3 Exploiting Reviews for recommender models

User reviews provide rich insights into user experiences, preferences, and opinions. Deep Cooperative Neural Networks (DeepCoNN) [25] learn item properties and user behaviors jointly from review text to potentially improve the quality of recommendations and alleviate the sparsity problem. They further enhance the model interpretability by exploiting rich semantic representations of review texts with CNNs. In their work, one of the CNN networks models user behavior using the reviews written by the user, and the other network models item properties using the written reviews for the item. Other works like Wu et al. [19] proposes a co-trained neural network with recurrent architecture for combining numerical ratings, natural language reviews, and temporal dynamics to achieve highly accurate recommendations. Tay et al. [15] proposed a review based recommendation system with multi-pointer co-attention.

In this work, we utilize item reviews to extract positive aspects of items and harness the encoding power of Knowledge Graphs to generate features for our items by linking aspects and other meta-information to train the GATNE model for generating relevant recommendations for anchor items along with their header texts. We highlight our methodology in detail in the next section.

3 Methodology

Our methodology revolves around training a Graph Neural Network (GNN) model to generate embedding representations for items,

which are then utilized to generate relevant items for a given set of anchor items. The following subsections present details of each component of the proposed DTS model.

3.1 Data Collection

E-commerce session data captures the user's journey through the website within a single browsing period. Analyzing co-bought and co-add-to-cart patterns within session data can reveal complementary item relationships. This strategy helps identify frequently purchased together items, potentially uncovering implicit purchase intentions and informing product bundling strategies. We restricted our data for a time frame of 3 months and top-5 performing product categories, as shown in 2. Within the product hierarchy, each broad product category is further subdivided into more granular classifications known as product types. For instance, the *Home & Garden* product category encompasses a variety of product types, including Garden Center, Grills & Outdoor Cooking, Indoor & Outdoor Plants, Outdoor Power Equipment, Pool Supplies, Patio Furniture, Flooring, Hardware, Kitchen Renovation, Paint, Tools, Wall Coverings etc. We construct the network G, where all item nodes are represented as $\mathcal{V}$, and edge types- *co-bought* and *co-add-to-cart* are represented as $\mathcal{E}$. We consider edges between items only if both items are from the same product category. If both items are bought in the session, they are connected by co-bought edge, if the items are added-to-cart in the same session, they are connected by co-add-to-cart edge. Our training data consists of two overlapping sets: 1) items interacted by customers on the e-commerce website, including items added to cart and transacted within the chosen time period, denoted as $\mathcal{V}_a$ and 2) items with extracted aspects from their reviews available on the website, denoted as $\mathcal{V}_b$. As these items possess a sufficient number of reviews suitable for aspect extraction, it is reasonable to categorize them as popular items. The anchor set, a subset of $\mathcal{V}_a$, consists of high performing items for which we generate a DTS recommendation carousel. The items of set $\mathcal{V}_b$ form the recall set of items. Our anchor set consists of 7 million nodes while the recall set consists of 1.1 million nodes. In the hierarchical classification of our product catalogue, the 1.1 million nodes with aspects are distributed according to Table 1.

Table 1. Summary of the product catalogue

Product Category	Number of items	Number of ptypes
Home & Garden	223,395	1539
Clothing, Shoes & Accessories	140,861	244
Books, Music & Movies	126,522	12
Health & Beauty	107,009	583
Electronics	49,316	346

3.2 Items' Aspects Extraction from User Reviews

Our proprietary ensemble linguistic model extracts these aspects and corresponding headers for each item from user reviews and assigns a relevance score to each aspect. This model primarily uses BERT-based aspect extraction, aspect-sentiment classification and aggregation. The goal of the aggregation step in the pipeline is to draw category-level and product-level insights based on all the individual opinions extracted within a category. In the aggregation, the following steps happen: Aspect unification (unifying different forms of aspects with the same meaning), Statistical validation (applying filters to remove outliers), Aspect naming (determining how to represent

Table 2.   Sample extracted aspects and corresponding headers for our product catalogue.

Item	Extracted Aspects	Headers
	• best gel • great for dry eyes • excellent ointment	• Customers say these are best gel • Similar items that are great for dry eyes • Customers say these are excellent ointment
	• great fit • comfortable • great cushioning • wear well	• Customers say these are a great fit • Customers say these are comfortable • Customers say these have great cushioning • Similar items that wear well

each aspect), Ontology creation (identifying the aspects hierarchy), Scoring (computing aspect scores per product). Aspect extraction is formulated as an iterative process. This approach leverages human intervention for labeling data and subsequent retraining of the aspect extraction BERT-based model. The 1.1 million nodes with aspects capture a total of 8354 distinct aspects, represented as $\mathcal{A}$. These aspects vary from- *great characters*, *great story* to *stylish* and *love the feel* to provide few examples. These aspects map to two header syntactic variations each starting with *Similar items* or *Customers say*. For example, *Similar items that are stylish* and *Customers say these are stylish* are two possible headers for the aspect - *stylish*. These headers are generated automatically, however evaluated manually for lexical clarity, conciseness, proper grammar, and overall readability. Finally, we have N_i extracted {aspects, relevance score} pairs as $\{a_{n_i}, asp\_rel_{n_i}\}$, where $(1 \leq n_i \leq N_i)$ for v_i item. $\mathcal{A}_{v_i}$ denotes the set of aspects for item v_i, i.e. $a_{n_i} \in \mathcal{A}_{v_i}$. Table 2 presents sample example of the extracted aspects and headers. Table 3 presents a summary of number of aspects per item. On an average, each item has 8 extracted aspects and corresponding headers.

Table 3.   Summary of number of aspects per item

mean	stddev	min	max
8.61	14.95	1	419

3.3   E-commerce Graph Creation

The items collected from the two overlapping sets, $\mathcal{V}_a$ and $\mathcal{V}_b$ are combined to construct a graph where nodes represent items, and edges represent customer interactions. Figure 2 shows a simplified e-commerce items' graph, where the item node shape represents the product category while the item node color represents the granular product type. Since edges are restricted to items from the same product category. G comprises of several sub-graphs corresponding to each product category. Each sub-graph has two edge types - *co-add-to-cart* edge type denotes adding items to the shopping cart while *co-bought* edge type denotes completing transactions.

3.4   Item Base Embedding

In training the GATNE model, two primary approaches exist for initializing node embeddings: random initialization and attribute-derived embeddings. Given our focus on aspects within the DTS
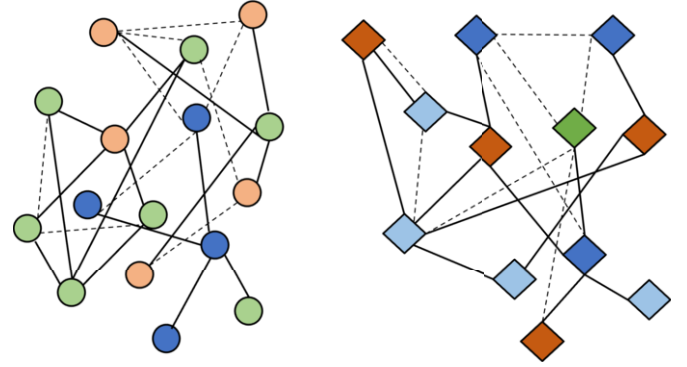
Figure 2.   Sample e-commerce graph, where item nodes are connected by *co-add-to-cart* (denoted by dashed lines) and *co-bought* (denoted by solid lines). Each sub-graph corresponds to each product category as edges are restricted among items of the same product type.

model, we opt for attribute-derived base embeddings. This selection allows the model to leverage the inherent meaning of item attributes during the embedding process. For the purpose of learning base embedding representations, we constructed a separate Knowledge Graph (KG) using the e-commerce item catalogue information and an existing item similarity model. Table 4 details the different tuples encoded for the knowledge graph training stage. These tuples leverage information extracted from various sources:

- **Item Catalog Data**: *Product type, Price Band*, and *Brand* relations are directly derived from the item catalog
- **Item Aspects Data**: The *Aspect* relation is established based on information described in Section 3.2
- **Proprietary Model**: The "Similarity" relation is constructed using our proprietary item-to-item similarity model.

Table 4.   Summary of the product catalogue

Head Entity	Relation	Tail Entity
Item	*Product type*	Aspect
Item	*Aspect*	Carousel Title
Item	*Similarity*	Item
Item	*Price Band*	Price Band Value (0-5)
Item	*Brand*	Brand popularity rank

DGL-KE (Deep Graph Library Knowledge Embedding) [24] is a framework specifically designed for representing and reasoning over

knowledge graphs within the Deep Graph Library (DGL). It facilitates the creation of knowledge graph embeddings by translating entities and relations into low-dimensional vector representations. This enables tasks like link prediction, entity classification, and knowledge graph completion within the DGL ecosystem. We leverage DGL-KE DistMult[22] to generate item base embeddings for our GATNE model. Our implementation closely follows the open source repository available on DGL website [6]. This approach allows us to encode rich relational information from the knowledge graph into low-dimensional vector representations of items, denoted as $x_i \in \mathbb{R}^{d_b}$. These learned item embeddings are then directly fed into the GATNE model as node **base embedding** for further processing and utilization within the network architecture.

3.5 Training GATNE model

We train the GATNE [1] model on the e-commerce graph, G, defined in section-3.3. The overall item embedding for each edge type comprises of two parts: **shared embedding** and **edge embedding**. The shared embedding of node v_i is shared between different edge types. In our graph representation, the edge embedding $\mathbf{u}_{i,r}^{(k)} \in \mathbb{R}^s$ for node v_i of relation type r at the k-th level ($1 \leq k \leq K$) aggregates the neighbors' edge embeddings using a mean aggregator [21] as:

$$\mathbf{u}_{i,r}^{(k)} = \sigma(\hat{\mathbf{W}}^{(k)} \cdot \text{mean}(\{\mathbf{u}_{j,r}^{(k-1)}, \forall v_j \in \mathcal{N}_{i,r}\})), \qquad (1)$$

where W is a learnable linear mapping matrix, σ is an activation function, s is the dimension of edge embeddings, K is the total aggregation levels and $\mathcal{N}_{i,r}$ contains the neighbors of node v_i on edge type r. The initial edge embeddings $\mathbf{u}_{i,r}^{(0)}$ for node i on edge type r is parameterized as the function of base embedding $\mathbf{x}_i$ as $\mathbf{u}_{i,r}^{(0)} = \mathbf{g}_r(\mathbf{x}_i)$, where $\mathbf{g}_r$ is a transformation function that transforms the base embedding to an edge embedding of node v_i on the edge type r. Following the aggregation process, we denote the final, K-th level edge embedding $\mathbf{u}_{i,r}^{(K)}$ for node v_i on relation type r as edge embedding $\mathbf{u}_{i,r}$. Subsequently, we concatenate all the edge embeddings associated with node v_i as $\mathbf{U}_i$ with size s-by-m, where s represents the individual edge embedding dimension and m represents the total number of relation types considered.

$$\mathbf{U}_i = (\mathbf{u}_{i,1}, \mathbf{u}_{i,2}). \qquad (2)$$

To effectively integrate information from all relevant edge types for a specific node, we leverage a self-attention mechanism [7] to get the coefficients $\mathbf{a}_{i,r} \in \mathbb{R}^m$ to compute of linear combination of vectors in $\mathbf{U}_i$ on edge type r as:

$$\mathbf{a}_{i,r} = \text{softmax}(\mathbf{w}_r^T \tanh{(\mathbf{W}_r \mathbf{U}_i)})^T, \qquad (3)$$

where $\mathbf{w}_r$ and $\mathbf{W}_r$ are trainable parameters for edge type r with size d_a and $d_a \times s$ respectively.
Overall Node embedding for node $\mathbf{v}_{i,r}$ for a given relation type r is computed as:

$$\mathbf{v}_{i,r} = \mathbf{h}(\mathbf{x}_i) + \alpha_r \mathbf{M}_r^T \mathbf{U}_i \mathbf{a}_{i,r} + \beta_r \mathbf{D}^T \mathbf{x}_i, \qquad (4)$$

where x_i represents the base item embedding learned from the DGL-KE model, $\mathbf{h}$ is a multi-layer perceptron [11] transformation function that transforms $\mathbf{x}_i$ to shared embedding. α_r, β_r are the hyper-parameter denoting the importance of shared embedding and edge embeddings respectively towards the overall embedding, set as 0.5 each based on empirical hyper-parameter tuning. $\mathbf{D} \in \mathbb{R}^{d_b \times d}$,

$\mathbf{M}_r \in \mathbb{R}^{s \times d}$ are the trainable transformation matrices.

Our model optimization leverages random walks to generate sequences of connected nodes within the graph. These node sequences are then employed as input for a skip-gram model [9, 10], which facilitates the learning of low-dimensional embeddings for the item nodes. We use negative log-likelihood and negative sampling to approximate the objective function as defined in [1].

Algorithm 1: Dynamic Text Snippets (DTS) Training

  Data: co-bought and co-add-to-cart items from e-commerce sessions data, item attributes data
  Input: e-commerce graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, embedding dimension d, random walk length l, learning rate η, negative samples N, coefficient α, β.
  Output: overall embedding $\mathbf{v}_{i,r}$ for all nodes for edge type $r \in \{$co-bought, co-add-to-cart$\}$
1 Model parameters (θ) initialization.
2 Base item embedding initialization as generated in section-3.4 Random walks generation: $\mathcal{W}_r$ for each edge type r.
3 Training samples generation: $\{(v_i, v_j, r)\}$ from random walks $\mathcal{W}_r$ for each edge type r.
4 **while** *not converged* **do**
5   **foreach** $(v_i, v_j, r) \in$ *training samples* **do**
6   Compute $\mathbf{v}_{i,r}$ using Equation (4)
7   Sample L negative samples and calculate skip-gram negative log-likelihood objective function.
8   Update model parameters θ.

3.6 Recall generation

To generate recommendations, an Approximate Nearest Neighbor (ANN) index is employed to retrieve the nearest neighbors for a given set of anchor items along with relevance score using FAISS [2, 4] *IndexFlatL2* index. For FAISS search, we restrict the recall set to items within $\mathcal{V}_b$. This strategic selection is motivated by the availability of extracted aspects for these items. By focusing on this subset, DTS recommendation carousel groups items that share same aspects, explained by the corresponding header. For each anchor item v_i, we retrieve 50 ANN recall set, denoted as $\mathcal{R}_i$ consisting of recall items $\{(r_i^p, faiss\_rel_i^p), (1 \leq p \leq 50)\}$, where $faiss\_rel_i^p$ is the FAISS relevance score for the corresponding recall item r_i^p. $\mathcal{A}_i$ is the set of unique aspects in the recall set $\mathcal{R}_i$.

3.7 Ranking, Aspect selection and Header Selection

With the anchor items and the corresponding recommendation set obtained from the ANN index, the most frequently occurring titles among the recommended items are extracted. These titles are then mapped to the recommendation list, creating a one-to-many mapping structure suitable for displaying recommendations in a carousel format on the website. Each anchor item v_i is associated with the recall set $\mathcal{R}_i$. $\mathcal{A}_{r_i^p}$ is the set of unique aspects for the recall item r_i^p. Header score for a given recall item r_i^p and aspect a pair is given as $faiss\_rel(r_i^p, v_i) \times asp\_rel(a, r_i^p)$. N_a represents the number of items returned in the recall set that share aspect a. Header score for a given aspect is computed as the average of the individual header scores assigned to all recall items having that aspect.

$$\mathbf{asp_{sel}} = \arg\max_{a \in \mathcal{A}_i} \left(\frac{1}{N_a} \sum_{\substack{r_i^p \in \mathcal{R}_i \\ a \in \mathcal{A}_{r_i^p}}} faiss\_rel(r_i^p, v_i) \times asp\_rel(a, r_i^p) \right)$$

(5)

Finally, the header with the highest aspect header score is chosen for DTS carousel as shown in equation-5.

Figure-3 shows the overall inference pipeline for the DTS model, where the final carousel shows the most relevant header along with corresponding recall items possessing that aspect.
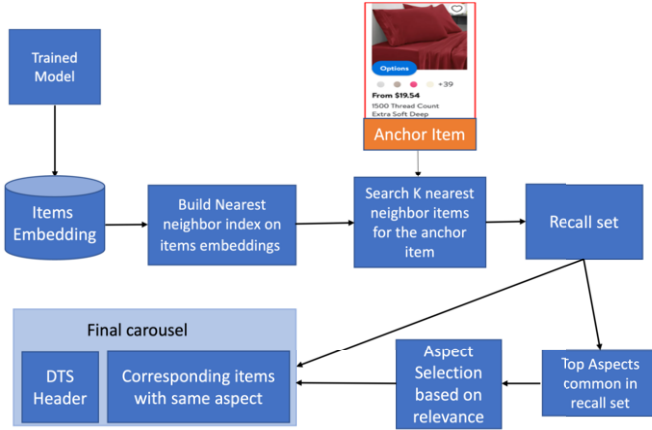

Figure 3. Dynamic Text Snippets (DTS) Inference Pipeline

4 Experiments and Results

4.1 Training Parameters

We utilized the DGL-KE open-source package [24] for training the DistMult model. The parameters chosen for the base embedding generation model are presented in Table 5.

Table 5. DistMult Training Parameters

Parameter	Value
batch size	1000
neg_sample_size	200
regularization_coef	1e-9
hidden_dim	128
gamma	19.9
lr	0.25

The GATNE model implementation closely resembles the DGL open source implementation [17]. To scale the graph for incorporating millions of nodes in the training process, we generate the random walks and training pairs separately and utilize these during training instead of creating walks every training iteration. The parameters for the best performing GATNE model are presented in Table 6.

4.2 Base Item Embedding Ablation Result

To comprehensively evaluate base item embedding representation strategies, we investigated the use of BERT-based embeddings and compared their performance with DGL-KE as inputs to the GATNE model. As *Home and Garden* is the product category with maximum

Table 6. GATNE Training Parameters

Parameter	Value
embedding_size (d, d_b)	128
embedding_u_size (s)	10
edge_type_count $(|\mathcal{E}|)$	2
negative samples (N)	4
neighbour_samples (for weight update)	[5]
dim_a (d_a)	20
size_of_window (for generating training pairs)	3
random_walk_length (l)	5
num_walks	5
lr	1e-3
optimizer	Adam

items that contain aspects, we created a subset of 30K items for experimenting with different base embedding representations. We used these base embedding variations, x_i to train the GATNE model, create DTS recall set and compare its performance against our website's best performing recommendation model. The results are reported in Table 7.

Table 7. Performance of different base embedding representations

Model	Overall NDCG Score
DistMult KE	0.50
Tiny Bert [13] (sentence-transformers/paraphrase-TinyBERT-L6-v2)	0.52

While linguistic BERT-based features were considered during our model prototyping, they were ultimately not chosen due to challenges in scaling the 768-dimension feature representations during model deployment on the available resources.

The NDCG score affirmed the relevance of our recommendations, however, due to the lack of a baseline model for comparison due to the novelty of our design, we decided to subject our model to an A/B test utilizing live traffic in order to ascertain its robustness and effectiveness.

4.3 A/B Test Result

We followed a standard GNN training paradigm in which we observed a minimization and saturation of loss values during the embedding generation step for the graph with 7 million nodes. Following this, our recall set was generated along with multiple titles for each anchor set.

We then initiated an A/B test involving millions of customers to evaluate the functionality of our model in a real-world setting. Initially, we limited traffic exposure to 5%, but upon observing satisfactory metrics, we expanded exposure to full traffic across all platforms, including Web, iOS, and Android. Our test was conducted for traffic on web pages of anchor items for which we had recommendations for the DTS carousel. While the control and variation shared two common carousels, there was a notable distinction in the third carousel, with the variation containing our DTS model as a recommendation carousel.

The A/B test lasted for two weeks with 100% traffic, during which time our model consistently demonstrated positive performance. We observed a GMV lift of 0.05% on Web, 0.07% on Android and 0.04% on iOS. As a result, it was subsequently launched on the website. This suggests that despite the lack of a conventional baseline model for comparison, our novel model has proven its effectiveness in a

live testing environment, thereby validating its utility and potential for wider application.

While delving deeper into the results, we observed a notable consistency across all platforms where the overall add-to-cart (ATC) feature remained neutral. However, there was an observable net increase in ATC and item clicks when calculated for the combination of these three carousels. This potentially indicated that users were locating relevant information more quickly and efficiently.

This outcome was uniform for Android, Web, and iOS platforms. When we dissected the data platform-wise, we observed that for Android, the highest net ATC changes were recorded in Media & Gaming, Electronics, and Sporting goods. However, Grocery and personal care exhibited a negative change which may suggest a need for improvement in these areas.

For both the Web and iOS platforms, Electronics, Grocery, and Home management were found to contribute significantly to the overall increase in ATC. These results highlight the nuances of user behavior and preferences across different platforms, and suggest targeted strategies might be necessary to optimize user engagement.

Owing to its overall positive performance, the DTS recommendation generation pipeline is successfully deployed and integrated into the live website, enabling dynamic headers for item recommendations captured by the graph representation.

5 Conclusion and Future Work

This study has highlighted the potential of utilizing user-generated reviews as a tool for creating recommender systems with dynamic text snippets (DTS). Our proposed model presents an encouraging method for improving the recommendation experience by integrating the aspects that users correlate with items. This approach can significantly contribute to more informed decision-making processes and enhance user satisfaction.

Our work has provided a solid foundation for future studies in this field. With a baseline model established, subsequent work will endeavor to incorporate other modelling techniques to improve the performance of aspect-based recommender systems further. One promising direction is the transition from dynamic text snippets (DTS) to personalized text snippets (PTS). This will entail mapping user affinities to aspects, thereby enabling us to recommend not only items but also header texts directly to users during their browsing sessions. While Generative AI offers exciting possibilities for dynamic header generation, the grounding problem remains a significant hurdle. This limitation arises because, in our context, the recommended items must have the chosen aspect for the generated header to be truly informative. Overcoming this grounding problem by incorporate item catalog knowledge graph and reviews reasoning capabilities should be another promising direction.

Through this strategy, we hope to create a more personalized and engaging user experience. This could potentially improve the way recommender systems function, making them even more responsive and tailored to individual user needs and preferences. We believe that our work paves way for exciting advancements in the field of recommendation systems and offers substantial opportunities for future contributions.

References

[1] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, and J. Tang. Representation learning for attributed multiplex heterogeneous network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1358–1368, 2019.

[2] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou. The faiss library. 2024.

[3] D. Jannach and M. Jugovac. Measuring the business value of recommender systems. *ACM Transactions on Management Information Systems (TMIS)*, 10(4):1–23, 2019.

[4] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[5] J. A. Konstan and J. Riedl. Recommender systems: from algorithms to user experience. *User modeling and user-adapted interaction*, 22: 101–123, 2012.

[6] A. Labs. DGL-KE documentation, 2020. URL dglke.dgl.ai/doc/.

[7] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[8] S. M. McNee, J. Riedl, and J. A. Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI'06 extended abstracts on Human factors in computing systems*, pages 1097–1101, 2006.

[9] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[11] S. K. Pal and S. Mitra. Multilayer perceptron, fuzzy sets, classifiaction. 1992.

[12] H. Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods, 2016. URL https://semantic-web-journal.net/system/files/swj1167.pdf.

[13] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL http://arxiv.org/abs/1908.10084.

[14] K. Tarnowska, Z. W. Ras, and L. Daniel. *Recommender system for improving customer loyalty*, volume 1. Springer, 2020.

[15] Y. Tay, A. T. Luu, and S. C. Hui. Multi-pointer co-attention networks for recommendation. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2309–2318, 2018.

[16] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[17] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

[18] Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE transactions on knowledge and data engineering*, 29(12):2724–2743, 2017.

[19] C.-Y. Wu, A. Ahmed, A. Beutel, and A. J. Smola. Joint training of ratings and reviews with recurrent recommender networks. 2017.

[20] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui. Graph neural networks in recommender systems: A survey. *ACM Comput. Surv.*, 55(5), dec 2022. ISSN 0360-0300. doi: 10.1145/3535101. URL https://doi.org/10.1145/3535101.

[21] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[22] B. Yang, W. tau Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases, 2015.

[23] S. Zhang, L. Yao, A. Sun, and Y. Tay. Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys (CSUR)*, 52(1):1–38, 2019.

[24] D. Zheng, X. Song, C. Ma, Z. Tan, Z. Ye, J. Dong, H. Xiong, Z. Zhang, and G. Karypis. Dgl-ke: Training knowledge graph embeddings at scale. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 739–748, New York, NY, USA, 2020. Association for Computing Machinery.

[25] L. Zheng, V. Noroozi, and P. S. Yu. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 425–434, 2017.