An Interpretable Ensemble of Graph and Language Models for Improving Search Relevance in E-Commerce

Nurendra Choudhary
Virginia Tech, Amazon
Arlington, VA, USA
nurendra@vt.edu

Edward W Huang
Amazon
Palo Alto, CA, USA
ewhuang@amazon.com

Karthik Subbian
Amazon
Palo Alto, CA, USA
ksubbian@amazon.com

Chandan K. Reddy
Virginia Tech, Amazon
Arlington, VA, USA
reddy@cs.vt.edu

ABSTRACT

The problem of search relevance in the E-commerce domain is a challenging one since it involves understanding the intent of a user's short nuanced query and matching it with the appropriate products in the catalog. This problem has traditionally been addressed using language models (LMs) and graph neural networks (GNNs) to capture semantic and inter-product behavior signals, respectively. However, the rapid development of new architectures has created a gap between research and the practical adoption of these techniques. Evaluating the generalizability of these models for deployment requires extensive experimentation on complex, real-world datasets, which can be non-trivial and expensive. Furthermore, such models often operate on latent space representations that are incomprehensible to humans, making it difficult to evaluate and compare the effectiveness of different models. This lack of interpretability hinders the development and adoption of new techniques in the field. To bridge this gap, we propose Plug and Play Graph LAnguage Model (PP-GLAM), an explainable ensemble of plug and play models. Our approach uses a modular framework with uniform data processing pipelines. It employs additive explanation metrics to independently decide whether to include (i) language model candidates, (ii) GNN model candidates, and (iii) inter-product behavioral signals. For the task of search relevance, we show that PP-GLAM outperforms several state-of-the-art baselines as well as a proprietary model on real-world multilingual, multi-regional e-commerce datasets. To promote better model comprehensibility and adoption, we also provide an analysis of the explainability and computational complexity of our model. We also provide the public codebase and provide a deployment strategy for practical implementation.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**; *Query representation*; • **Applied computing → Online shopping**.

KEYWORDS

Graphs, language models, search relevance, ensemble, plug and play, e-commerce, query

WWW '24 Companion, May 13–17, 2024, Singapore, Singapore
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0172-6/24/05.
https://doi.org/10.1145/3589335.3648318

ACM Reference Format:
Nurendra Choudhary, Edward W Huang, Karthik Subbian, and Chandan K. Reddy. 2024. An Interpretable Ensemble of Graph and Language Models for Improving Search Relevance in E-Commerce. In *Companion Proceedings of the ACM Web Conference 2024 (WWW '24 Companion), May 13–17, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3589335.3648318

1 INTRODUCTION

The rapid advancements in language modeling and graph neural network research have created challenges for traditional industry frameworks looking to adopt the latest developments. Adopting a new model requires extensive experimentation to determine its generalizability on practical datasets, but the ever-changing nature of these datasets makes it difficult to compare old models with new ones. As a result, all models and datasets must be carefully considered before determining if a new model can be adopted in practice. This presents a significant challenge for industry frameworks seeking to incorporate the latest advancements in the field. In this study, we focus on the specific issue of search relevance[1] in the e-commerce industry, where the goal is to classify the relationship between a query and a product as one of exact, substitute, complement, or irrelevant (ESCI), as shown in Figure 1. Human annotation for the purpose of training classification models in the domain of e-commerce search (ESCI) can be a resource-intensive task. The datasets that are typically used for training these models tend to be significantly smaller than those used for training relevance models, which often rely on anonymous, aggregated customer shopping behavior data. This is particularly true for labeled data that is available for training on a per-region basis, which tends to be even smaller in size. However, e-commerce datasets do contain additional information that may be useful for model training, such as query-item graphs that capture interactions between queries and items, and co-purchase behavior across products in a catalog. At first glance, search relevance may appear to be a straightforward multi-classification task. However, the short text nature of queries and products [4] as well as the influence behavioral signals have on their relationship make it a non-trivial problem.

Initial work in this field [21, 25] relied on the semantic features of queries and products, but subsequent research has highlighted the importance of incorporating behavioral signals [5] such as click-through data and purchase data in learning representations. However, these methods combine semantic and graph information in a latent space, which results in a lack of interpretability. Furthermore, they are standalone models that must be individually trained and evaluated for their specific scenario. This approach faces three

[1]We chose search relevance due to the practical impact of the problem and the availability of public data. Our proposed methods are extensible to other problems.

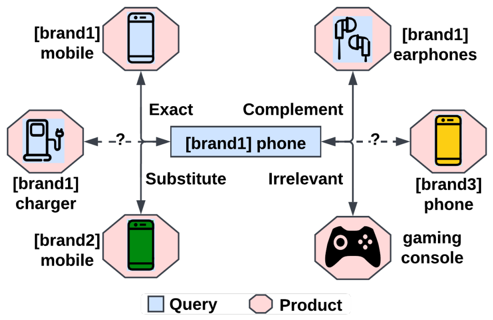


Figure 1: An illustration of the search relevance problem. Using the semantic information of queries and products along with the behavioral relationships between them, the goal is to classify the degree of relevance for unseen query-product pairs (labeled with question marks) as exact, substitute, complement, or irrelevant.

major challenges: (i) the dynamic nature of real-world data requires frequent and costly updates to the deployment model, often requiring the retraining of all previous candidates to maintain comparability, (ii) certain decisions, such as which semantic/behavioral signals to include, cannot be automated, and (iii) model operations in a latent space result in limited interpretability of the results.

To overcome these challenges, *we propose Plug and Play Graph LAnguage Model (PP-GLAM), a modular ensemble of LMs and GNNs* that relies on additive explanation values to automatically decide on the inclusion of semantic and behavioral signals. Our model uses an ensemble of LMs [13] to capture semantic information from the query and product. While previous approaches [6, 25] also utilize LMs, they generally default to a multilingual framework due to the multilingual nature of e-commerce datasets. However, LMs specifically developed on resource-rich languages such as English tend to outperform their multilingual counterparts [16]. Thus, it is advantageous to combine the benefits of language-specific models with multilingual ones. Consequently, in PP-GLAM, we encode the query-product semantic information using multiple language-specific and multilingual LM models, and rely on a Gradient-Boosted Decision Tree (GBDT) for effective model selection. However, a majority of the language models only support a fixed number of tokens, which are not sufficient for handling long product descriptions. While recent works such as Longformers [45] are able to circumvent this issue, they are computationally more expensive and add to both inference time and cost. Thus, in our model, we denoise the product description to a fixed token size using certain importance measures [31]. More importantly, we also leverage several query-product signals such as clicks, purchases, add-to-carts (or adds), impressions, and consumes. These relations vary both in their density and correlation with target variables (e.g., clicks are more common than purchases, but purchases are more correlated to a query-product match). This density-correlation trade-off of relations must be thoroughly studied for every task and deployment iteration, which adds to the deployment time and effort. To alleviate this problem, we automate these decisions by considering all the potential homogeneous (single-relation) and heterogeneous (multi-relation) GNNs in a GBDT ensemble, subsequently eliminating the worst-performing models using their ranking over explainable SHAP values [20]. Unlike MLP [28] and Attention models [39], which aggregate features in a latent space, our GBDT based ensemble can be made explainable using additive SHAP values. The additivity allows us to independently decide the importance of individual LMs and relational GNNs on the target variable and plug-and-play them to improve model performance. Subsequently, automating the decision of including the best-performing models can be done based on the computational constraints. We conduct extensive experiments on large-scale e-commerce data to demonstrate that our ensemble framework significantly outperforms the state-of-the-art baselines in the e-commerce tasks of search relevance and irrelevant detection, respectively. Furthermore, we conduct several studies to analyze the computational complexity of our model. We also demonstrate our model's interpretability through reports of the target variable's dependence on the models and relations. Lastly, we provide a deployment strategy to apply our modular framework in a practical environment. To summarize, our main contributions are as follows:

(1) In this paper, we present PP-GLAM, a modular ensemble of LMs and relational GNNs that utilizes GBDTs for flexible model selection in a practical environment.
(2) We demonstrate the effectiveness of PP-GLAM on search relevance and irrelevant detection compared to state-of-the-art public and proprietary baselines on a multi-regional e-commerce dataset.
(3) We show the benefits of utilizing ensembles as an interpretable strategy to aggregate semantic and behavioral signals and efficiently select the most impactful models.
(4) We provide a detailed strategy to deploy our modular framework in a practical environment with dynamic data sources.

The rest of the paper is organized as follows: Section 2 discusses the relevant background and Section 3 describes our proposed model architecture. Section 4 demonstrates our experimental results and subsequent analysis. Section 5 details our deployment strategy and Section 6 concludes our work.

2 RELATED WORK

In this section, we discuss previous work related to our problem in the areas of language models, graph neural networks, and graph-enhanced language models.

Language Models (LMs). Transformer-based LMs are a type of neural network architecture that has revolutionized the field of natural language processing (NLP) in recent years. These models have achieved state-of-the-art performance on a wide range of NLP tasks such as machine translation [1], language generation [3], and search [8]. The development of transformer-based LMs originated from the introduction of word embedding models, such as Word2Vec [22] and GloVe [29], which represent words as continuous vectors in a high-dimensional space to capture the semantics and relationships between words. These embeddings have been used as inputs to deep learning models, such as recurrent neural networks (RNNs) [37] and convolutional neural networks (CNNs) [35]. However, such methods are limited in their ability to capture word sense and contextual semantics. Transformer-based models [39] were introduced in conjunction with RNNs and CNNs to capture the contextual dependencies between words in a sequence. However, further research [39] demonstrated their standalone effectiveness and led to the rise of task-tunable pre-trained LMs such as BERT [8] and XLNET [42]. Recent developments in LMs have focused on multilingual extensions [7] and increasing the size and complexity of the models. Given this rapid pace of advancement, it is important that industry frameworks can quickly adopt and deploy the latest LMs. Thus, we design PP-GLAM as a flexible modular ensemble.

Graph Neural Networks (GNNs). GNNs are a special type of neural network architecture, designed to operate on graph-structured data. Early research in graph processing predominantly relied on matrix factorization [27, 33] and random walk-based [9, 23, 38] approaches. Matrix factorization is a technique that utilizes linear algebra techniques [18, 26] to simplify the representation of the relationships between nodes in a graph by decomposing the adjacency matrix into a lower-dimensional, latent space. Random walks use Markov chain models to simulate transitions between nodes and learn their representations. These techniques demonstrate the effectiveness of vector space models in graph representation learning. However, they are limited in their ability to capture node neighborhood relations. To address these limitations, GNNs were adopted to effectively aggregate node neighborhood features, typically for node classification [34], link prediction [17], and graph classification [24]. These techniques include Graph Convolutions (GCNs) [17], Graph Attention (GAT) [40], and GraphSage [10], which respectively use convolution filters, attention mechanisms, and graph sampling to learn node and link representations for downstream tasks. Current research in the area focuses on applying GNNs to heterogeneous graphs [41], temporal graphs [32], and knowledge graphs [2]. Given the short text nature of queries in the task of search relevance, we need to utilize behavioral graph signals to better understand users' intent. However, with the diversity of behavioral signals available in e-commerce datasets, it is difficult to select the most significant candidates for a downstream task. Therefore, our model utilizes SHAP values to select the set of best-performing signals and corresponding GNNs in a computationally constrained deployment environment.

Graph-Enhanced Language Models (GELM). Recently, researchers have focused on integrating graph signals and text features in unified GELM models. Initial studies, such as TextGNN and TextGCN, used semantic features to initialize GNN models for downstream tasks such as node classification and link prediction. However, a primary limitation of this approach is its limited ability to learn task-specific semantic embeddings. To overcome this issue, Graphormers [43] combined manual graph features, such as spatial encoding and centrality measures, with a text-based Transformer architecture to capture hybrid features. This allows the model to effectively handle both graph and text data. However, calculating the required graph features is computationally expensive and requires frequent model re-training with dynamic datasets. To avoid this, the (proprietary) SALAM model [5] uses an attention mechanism to combine graph and text features in a scalable industry setting. Furthermore, GreaseLM [46] employs intermediate MLP units after each layer to combine graph (GNN layer) and text (LM layer) features. However, aggregations in such models occur in a latent space, and are hence not interpretable for decision-making. Thus, *we developed PP-GLAM for industry settings as a modular combination of LMs and relational GNNs that use interpretable SHAP values to decide on the model components.*

3 PROPOSED FRAMEWORK

This section discusses the problem statement of the search relevance task, describes the different components of our model, and explains its training and inference pipeline.

3.1 Problem Statement

Let the set of query-product pairs be (Q, P) and the corresponding query-product graphs of clicks, impression, adds, purchases, consumes, and a combined heterogeneous version (any) be denoted by their adjacency matrix $\mathcal{G}^{\psi} : Q \times P$, where signal $\psi \in$ {clicks, impressions, adds, purchases, consumes, any}. Note that, "any" signal implies the presence of any of the mentioned relations. Each element G^{ψ}_{qp} indicates the rate of signal ψ between query $q \in Q$ and product $p \in P$ (e.g., the number of product clicks for a particular query). The objective of search relevance is to learn a model P_θ with parameters θ that classifies query-product pairs into the following relevance classes: exact (E), substitute (S), complement (C), and irrelevant (I). The model is formulated as:

$$\hat{y} = \underset{y'=\{E,S,C,I\}}{\arg\max} \; P_\theta(y'|x,\theta); \quad \theta = \underset{\theta'}{\arg\max}\, P_{\theta'}(\hat{y} = y|x,\theta') \tag{1}$$

where θ' is a sample parameter set from the parameter search space, $y, \hat{y} \in \{E, S, C, I\}$ are the ground-truth value and the model output from P_θ for an input $x = \left(q_i, p_j, G^{\psi}\right)$, respectively.

3.2 Model Elements

This section describes the different elements of our model and the ensemble of language and graph features for better generalization over diverse e-commerce datasets. Our model pipeline starts with the input query $x = (q, p)$ with which the corresponding subgraph $\mathcal{G}^{\psi}(q, p)$ is constructed using the Graph Extraction module. Subsequently, the textual information and subgraph are encoded using multiple language models $\{LM_i(x)\}_{i=1}^{|LM|}$ and relational GNNs $\{GNN_j(\mathcal{G}^{\psi}(q, p))\}_{j=1}^{|GNN|}$ to produce the corresponding ESCI labels $\hat{Y} = \left\{\hat{y}^{LM}_1, \hat{y}^{LM}_2, ..., \hat{y}^{LM}_{|LM|}, \hat{y}^{GNN}_1, \hat{y}^{GNN}_2, ..., \hat{y}^{GNN}_{|GNN|}\right\}$. These label outputs $\hat{Y}$ and certain manual features (such as language information) are finally ensembled together in a GBDT model to produce the final PP-GLAM output.

Graph Extraction. In this module, we extract the local neighborhood subgraph pertaining to the input query-product pair $x = (q, p)$. In our problem, we use the signals of $\psi \in$ {clicks, impressions, adds, purchases, consumes, any}, with a corresponding adjacency matrix $\mathcal{G}^{\psi}$ for each signal. To construct the local k-hop neighborhood subgraph $\mathcal{G}^{\psi}(q, p)$, we perform a breadth-first traversal around both q and p until a maximum depth of k. We observe that this process is independent for each (q, p) pair, and hence, we parallelize it over the CPU cores and store the subgraphs as hash maps for constant time retrieval in the training and inference pipelines.

De-noising Product Information. Typical product data consists of long text descriptions that are not supported by generic language models. Hence, we utilize a TF-IDF [31] pre-processor

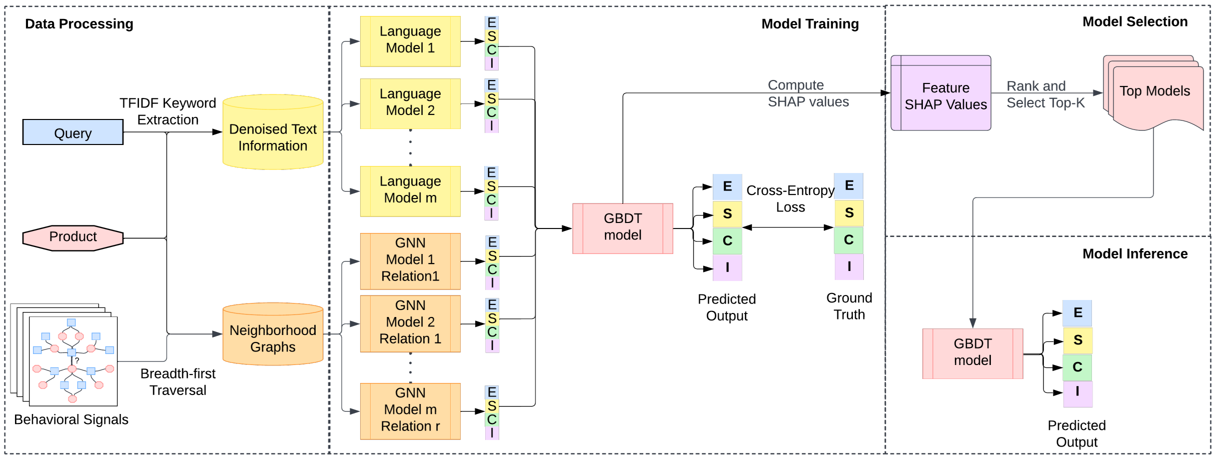


Figure 2: An overview of the PP-GLAM architecture. The model contains four modules: (i) Data Processing handles the pre-computation of graph neighborhoods and the de-noising of the product information, (ii) Model Training uses the training samples with query-product pairs and corresponding ground-truth labels to learn the parameters of the GBDT-based ensemble containing different language models and graph neural networks (GNNs), (iii) Model Selection module utilizes interpretable SHAP values to eliminate low-performing models based on the constraints of the inference setup, and (iv) Model Inference loads the selected set of models into memory for inference on new batches.

to rank the terms' importance in the description and eliminate words beyond the language model's supported capacity γ. This is formalized for a text sequence of tokens $\{t_i\}_{i=1}^{|p|} \in p$ as:

$$TFIDF(t_i, p) = TF(t_i, p) \times \log\left(\frac{|P|}{DF(t_i, P)}\right) \tag{2}$$

where $TF(t_i, p)$ is the number of times t_i occurs in p and $DF(t_i, P)$ is the number of $p \in P$ that contains t_i. Letting $TFIDF(t_\gamma, p)$ be the γ-th largest value in the sorted set of $\{TFIDF(t_i, p)\}_{i=1}^{|p|}$, then the filtered set $p' = \{t_i \in p | TFIDF(t_i, p) \geq TFIDF(t_\gamma, p)\}$.

The query and product information is then merged with separator tokens as $t = q \| [SEP] \| p'$, processed through the cross-encoder tokenizers [36], and stored as hash maps for efficient retrieval in the model pipelines.

Language Models. E-commerce datasets are generally diverse since they typically span multiple geographical regions with multiple languages. Hence, in our model, we utilize an ensemble $\{LM_i\}_{i=1}^{|LM|}$ of language-specific LMs that work well for resource-rich languages (English), including DeBERTa [13], COCOLM [44], BigBird [45], and a multilingual LM - M-DeBERTa [12]². The tokenized input sequence t is encoded using the LM encoders $LM_i \in LM$ into its encoding $e \in \mathbb{R}^d$ with a softmax layer $\phi_d^l : \mathbb{R}^d \rightarrow \mathbb{R}^l$ to obtain the probability over l output labels. The encoder is formalized as

$$\hat{y}_i^{LM}(t) = \phi_d^l\left(e_i(t)\right), \quad \text{where } e_i(t) = LM_i(t) \tag{3}$$

$$\hat{Y}_{LM}(t) = \left\{\hat{y}_i^{LM}(t) \middle| i \in [1, |LM|]\right\}, \quad \text{where } \hat{y}_i^{LM}(t) \in \mathbb{R}^l \tag{4}$$

Graph Models. To aggregate the structure signals, we use an ensemble of $|\psi|$ behavior signals over the GNN networks of GCN, GAT, and GraphSage². For an input query-product graph with v nodes and adjacency matrix $\mathcal{G}^\psi(q,p) \in \mathbb{R}^{v\times v}$, let the feature matrix be $h_0 \in \mathbb{R}^{v\times d}$ and (i_q, i_p) be the indices of query and product node in the feature matrix, then K-layer GNN models aggregate the node neighborhood information as:

$$h_{k+1} = \sigma\left(D^{\zeta-1}\mathcal{G}^\psi(q,p)D^\zeta h_k W_k\right), \quad \text{where } \zeta \in [0,1] \tag{5}$$

$$\hat{y}_i^{GNN}(h_0) = \phi_d^l(h_{k+1}[i_q] \| h_{k+1}[i_p]), \text{ where } \hat{y}_i^{GNN}(h_0) \in \mathbb{R}^l \tag{6}$$

$$\hat{Y}_{GNN}(h_0) = \left\{\hat{y}_i^{GNN}(h_0) \middle| i \in [1, |GNN|]\right\} \tag{7}$$

where ζ is the factor of spectral filter dependent on the GNN model, $D \in \mathbb{R}^{v\times v}$ is the diagonal degree matrix and $\|$ is the concatenation operator. Note that we solve the search relevance as a link prediction task in GNNs and hence only use (i_q, i_p) in the final softmax layer to compute the probabilities over l output labels.

Interpretable Ensemble. To aggregate the features from different models in an interpretable manner, we utilize Gradient Boosting Decision Trees (GBDTs) [15]. GBDTs are a type of ensemble machine learning algorithm that combines the predictions of multiple decision trees to improve the accuracy and stability of the model. The algorithm works by adding decision trees to the model sequentially, with each tree correcting the errors made by the previous tree. The trees are trained using gradient descent optimization, which minimizes the error between the predicted output and the ground truth. The final prediction is obtained by averaging the predictions of all trees in the ensemble. For an input text t, graph h_0, and associated region information f, the output of a GBDT model is formulated as:

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} o_t(\hat{Y}), \quad \text{where } \hat{Y} = \hat{Y}_{LM}(t) \| \hat{Y}_{GNN}(h_0) \| f \tag{8}$$

where $\hat{y}$ is the final prediction, T is the number of trees in the ensemble, and $o_t(\hat{Y})$ is the prediction of the t-th tree. In our task, we optimize the GBDT's model parameters θ by minimizing the

²Note that we only listed the LM and GNN models used in our experiments. The framework, however, supports most generic LM and GNN models.

cross-entropy loss for l-labels $Y = \{y_1, y_2, ..., y_l\}$ as follows:

$$L(y, \hat{y}) = -\sum_{y \in Y} (ylog(\hat{y}) + (1-y)log(1-\hat{y})) \tag{9}$$

$$\theta = \theta - \alpha \nabla_\theta L(y, \hat{y}) \tag{10}$$

We employ SHAP (SHapley Additive exPlanations) [20] to additively interpret the contribution of each feature to the GBDT ensemble. This helps us in model selection to add and eliminate LMs and GNNs in a constrained practical application. SHAP calculates the features' contribution using a recursive algorithm that traverses the decision tree and aggregates the contributions of every feature at each node. The SHAP values for a feature are the average change in the prediction caused by that feature, averaged over all possible ways that the feature could be used in the decision tree. For GBDTs, the SHAP value of a feature i is calculated as:

$$\Omega_i = \sum_{F \subseteq \omega_X} \frac{|F|!(|X| - |F| - 1)!}{|X|!} (\Omega_{F \cup i} - \Omega_F) \tag{11}$$

where Ω_i is the SHAP value for feature i, F is a subset of features, X is the set of all features, and ω_X is the set of all possible subsets of X. The SHAP values enable us to rank a model's contribution and use it for model selection. Note that, due to the additive nature of SHAP values, we can compute a model's contribution without re-training all previous candidates.

3.3 Model Components

Computational constraints significantly vary in the training and inference phase in industry settings. The training phase optimizes for model accuracy and generalizability over broader datasets, whereas the inference phase must additionally optimize for model latency under constrained computational resources[3]. In this section, we explain our model's training phase, model selection setup, and inference phase for a practical environment.

Training Phase. During the training phase, we have access to a significant amount of both CPU and GPU memory. Thus, we can optimally parallelize our data processing and model training steps. Additionally, we note that e-commerce queries have a $20\% - 25\%$ monthly update rate, i.e., we receive only $20\% - 25\%$ new queries every month while the rest overlap with previously processed queries. Thus, given the monthly cycle of model update, the time-intensive operations of Graph Construction and De-noising Product Information do not significantly affect the model pipelines in practice. However, we need to ensure that the models are compliant with industry standards, and thus, we optimize our hyperparameters (provided in Section 3.4) in a restricted search space (model size limited to 2.5 GB in our case). The training phase receives an input query-product pair with its corresponding relevance label. The input pair is processed using the graph signals and input to update the parameters of candidate LMs and GNNs through training. The final trained models are stored as checkpoints. Algorithm 2 in Appendix A presents the model's training flow.

Model and Relation Selection. In this phase, our goal is to restrict the number of candidate LMs and relation-based GNN models based on the inference constraints, while preserving the best performance. For an inference metric Λ (additive units such as GPU VRAM, CPU RAM, or inference time) with constraint K and the set of candidate models $\{LM\}_{i=1}^{|LM|}$ and $\{GNN\}_{j=1}^{|GNN|}$, this phase selects candidates $M \subseteq LM \cup GNN$, with the following constraints:

$$\sum_{m \in M} \Lambda_m \leq K; \quad \sum_{m \in M} \Omega_m \geq \sum_{f \in F-M} \Omega_f \tag{12}$$

We need to handle the selection in two scenarios: (i) initial setup where $M = \emptyset$ and (ii) continuous deployment where $M \neq \emptyset$. For the initial setup, we rank the candidate models according to their SHAP values (computed using Eq. (11)) and incrementally add them to M till the constraints hold. In the case of continuous deployment, we follow a last-in, first-out (LIFO) strategy and compare the candidate model f against the last-inserted model $m_l \in M$. We use the new candidate only if $\Omega_f > \Omega_{m_l}$ and $\sum_{m \in M - \{m_l\} + \{f\}} \Lambda_m \leq K$. The model selection algorithm is provided in Appendix A.

Inference Phase. In the inference phase, the model must handle a large volume of query-product pairs, and hence, the scalability and latency become crucial considerations. To address these concerns, we relegate the Graph Computation and De-noising Product Information modules to the pre-computation step. Thus, during inference, we load PP-GLAM parameters on the available GPUs and process the query-product pairs in parallel batches (retrieving the graph and de-noised product information in constant time). One potential issue with this method is that pre-computed graph and de-noised product information may not be available for infrequent query-product pairs. However, in typical e-commerce settings, we have observed that this information is available for approximately 80%-85% of queries each month. For the remaining queries, we can still use a subset of candidate models (which do not rely on graph information) and obtain suboptimal results even with incomplete or noisy product information. It should be noted that, because the candidate models' predictions are independent, we can set up reliability-availability trade-offs[4] according to the downstream use case. Our model's inference pipeline is presented in Appendix A.

3.4 Implementation Details

PP-GLAM is implemented using the PyTorch framework with the huggingface library for the LM models and the PyG library for the graph neural networks. The model is trained on sixteen Nvidia V100 GPUs and optimized using AdamW [19] with standard beta parameters of 0.9 and 0.999, weight decay rate of 0.01, and mini-batch gradient descent of batch size 64. The language models encode the input tokens with a maximum length of $|t| = 512$ into a $d = 768$-dimensional vector to be classified into $l = 4$ labels. For GNNs, we use $|\psi| = 5$ behavioral signals in a $K = 2$-layer GNN network with $l = 4$ final output labels. The graph extraction module obtains the local $k = 2$-hop neighborhood with a maximum size of 100 neighbors. The GBDT ensemble is implemented using the LightGBM library [15] with a cross-entropy objective, 1500 iterations, 0.005 learning rate, 15 leaves, 15 depth, and 200 bagging frequency. The inference setup is constrained to four V100 GPUs, with a time limit of 15 milliseconds per query-product sample. To avoid loading overheads, we use LuaJIT compiler to load PP-GLAM's parameters

[3]Complete details of the computational setup are presented in Section 3.4.

[4]A focus on reliability would imply all models succeed, whereas, availability is possible if any model succeeds.

for inference. The implementation code of our model is available here[5]. The dataset of query-product pairs is publicly available [30].

4 EXPERIMENTAL RESULTS

Our experimental setting investigates the performance of our model for the ESCI classification problem and analyzes its components in practical environments. More specifically, we study the following research questions (**RQs**):

RQ1. How does PP-GLAM's performance compare against current alternatives on the search relevance tasks of ESCI classification and irrelevant classification?
RQ2. Is PP-GLAM compatible with industry constraints? How does the performance differ in such scenarios?
RQ3. Is the ensemble method of feature aggregation better than other techniques?
RQ4. How do we make model decisions using the additive SHAP values?

4.1 Datasets and Baselines

For our experiments on search relevance tasks, we utilize the publicly available ESCI dataset [30]. The dataset contains ≈2 million real-world query-product pairs with manually annotated ESCI labels, collected from the regions of the United States (US), Japan (JP), and Spain (ES). The regions are diverse in their languages, geographical location, and user behavior. This dataset is enriched with five additional behavioral signals: impressions (user viewed the product), clicks (user clicked on the product), adds (user added the product to their cart), purchases (user purchased the product), and consumes (user finally consumed the product). As shown in Figure 3, we found that certain behavioral signals such as adds and purchases have a strong correlation with target ESCI labels, whereas others such as impressions and clicks have a denser availability. The dataset does not contain any customer information or personally identifiable information. From the perspective of an e-commerce search engine, we evaluate our model on two real-world tasks of ESCI classification (applied in complementary product recommendation [11]) and irrelevant classification (used for improving user experience by increasing the precision of product recommendation). Each task leads to a unique imbalanced class distribution, which our model handles with class weights in the loss function. For the experiments, we hold out the test set (10% of the dataset) for evaluation and perform a five-fold cross-validation with a train-validation split of 4:1. Further details on the datasets' edge distribution and edge correlation are provided in Table 1 and Figure 3, respectively. Note that densely available behavioral signals such as impressions and clicks have a low correlation to the target ESCI labels when compared to sparser signals such as clicks and adds. Therefore, in practical applications, it is crucial to carefully weigh the trade-off between reliability and availability when selecting which signals to use. The dataset for the query-product pairs is available here[6].

For the baselines, we select (i) the popular language models of DeBERTa, COCOLM, BigBird, and M-DeBERTa, (ii) GNN models of GCN, GAT, and GraphSage, (iii) Relation-GNN models of GraphSage model trained using behavioral signal-attributed edges[7], and (iv) SALAM [5], which is a graph-based language model for e-commerce engines. Language models only utilize the query-product text information. For graph models, the node features are initialized using semantic embeddings from LM models (M-DeBERTa). SALAM utilizes both the text information as well as the behavioral signals in its framework.

Table 1: Distribution of the edges in the dataset. The columns represent the number of edges and its proportion in the overall dataset.

Relation	Number	Proportion (%)
impressions	30,256,494	40.15
clicks	38,739,273	49.86
adds	3,886,923	5.25
purchases	1,445,229	1.93
consumes	118,440	0.17
exact	1,451,542	1.72
substitute	487,544	0.58
complement	64,481	0.08
irrelevant	223,804	0.27
Total	76,673,730	100

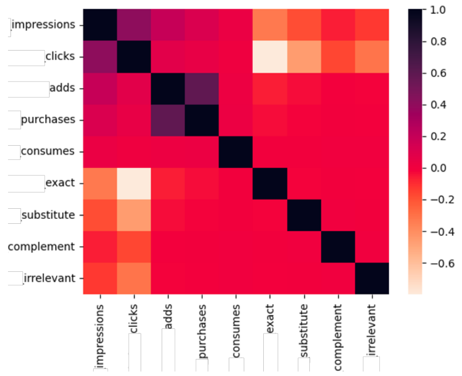


Figure 3: Correlation between edges of the dataset. Note that dense signals such as impressions and clicks have a low correlation with target edges of exact, substitute, complement, and irrelevant, whereas sparser signals such as adds and purchases are highly correlated. Hence, we must carefully consider the reliability-availability trade-off in practice.

4.2 RQ1: Search Relevance

In this experiment, we investigate the performance of our model relative to other baselines on the search relevance tasks of ESCI classification and irrelevant classification. The PP-GLAM model predicts a single label for each query-product pair and the performance is evaluated using ground-truth labels on the metrics of accuracy, macro-F1, and weighted F1 scores.[8] We evaluate both the macro-F1 and weighted F1 scores due to class imbalance, and certain downstream tasks such as recommendation systems require the weighted F1 to be high. For other tasks, such as irrelevant detection, it will be ideal to obtain a high macro-F1.

From our experimental results, reported in Table 2, we observe that our model consistently outperforms the baselines in both the tasks of ESCI classification and irrelevant classification. Among the

[5]https://github.com/amazon-science/graph-lm-ensemble
[6]https://github.com/amazon-science/esci-data

[7]the basic relations are impressions, adds, clicks, purchases, and consumes. However, we also use a homogeneous (All) and heterogeneous (Het-All) version of the graph with all the basic relations.
[8]weights are given by the number of actual occurrences of the class in the dataset.

Table 2: Performance comparison of PP-GLAM against baselines for search relevance tasks of ESCI classification and irrelevant classification in the e-commerce regions of US, ES, and JP. Evaluation metrics (presented in the columns) include Accuracy (Acc), Macro-F1 (MacF1), and Weighted-F1 (WtF1). The best results for each metric are in bold font. All the improvements of PP-GLAM over SALAM and best-performing models are statistically significant with a p-value threshold of 0.05.

Task		ESCI Classification									Irrelevant Classification								
Model	**Locale**	**United States (US)**			**Spain (ES)**			**Japan (JP)**			**United States (US)**			**Spain (ES)**			**Japan (JP)**		
Type	**Model**	**Acc**	**MacF1**	**WtF1**	**Acc**	**MacF1**	**WtF1**	**Acc**	**MacF1**	**WtF1**	**Acc**	**MacF1**	**WtF1**	**Acc**	**MacF1**	**WtF1**	**Acc**	**MacF1**	**WtF1**
LM	**(M)DeBERTa**	84.42	70.00	84.42	76.84	66.99	76.84	75.91	64.33	75.91	95.71	87.10	95.61	86.52	46.39	80.27	90.20	81.52	90.18
	COCOLM	81.19	64.19	81.19	-	-	-	-	-	-	95.18	84.58	94.91	-	-	-	-	-	-
	BigBird	86.58	74.40	86.58	-	-	-	-	-	-	95.92	90.96	95.89	-	-	-	-	-	-
GNN	**GraphSage**	75.29	46.91	71.85	63.79	40.61	59.23	64.67	40.93	60.84	90.41	47.49	85.87	86.52	46.41	80.27	86.22	46.45	79.86
	GCN	71.47	34.53	64.15	60.34	34.78	55.46	60.99	35.25	55.28	90.40	47.49	85.85	86.54	46.39	80.28	86.20	46.42	79.82
	GAT	72.14	38.99	66.75	60.65	35.60	56.13	61.92	37.23	57.22	90.43	47.49	85.86	86.53	46.39	80.27	86.22	46.44	79.85
Relation	**Impressions**	72.38	37.80	66.06	61.49	37.07	56.76	62.56	36.95	57.12	90.40	47.51	85.84	86.53	46.40	80.30	86.21	46.41	79.84
GNN	**Adds**	73.56	41.95	68.81	61.62	37.57	57.51	62.79	39.15	58.89	90.41	47.48	85.87	86.53	46.40	80.30	86.20	46.46	79.85
	Clicks	72.43	38.16	66.19	61.37	37.15	56.78	62.51	36.70	56.90	90.42	47.48	85.85	86.53	46.40	80.30	86.21	46.42	79.83
	Purchases	73.92	43.27	69.76	61.80	37.19	57.34	62.84	39.22	58.93	90.43	47.49	85.85	86.53	46.41	80.29	86.22	46.44	79.86
	Consumes	73.90	44.81	70.72	61.88	36.75	57.21	62.21	39.75	59.26	90.43	47.50	85.86	86.53	46.40	80.29	86.20	46.51	79.87
	All	72.22	37.23	65.59	61.45	36.91	56.64	62.37	36.26	56.54	90.43	47.48	85.85	86.53	46.41	80.28	86.21	46.39	79.83
	Het-All	72.35	37.80	65.94	61.52	36.35	56.33	62.43	36.59	56.83	90.42	47.49	85.84	86.55	46.40	80.28	86.21	46.44	79.85
GNN	**SALAM**	83.82	67.17	84.61	75.77	60.72	76.49	75.74	63.71	74.51	89.82	73.17	90.61	81.77	44.72	78.49	86.74	70.71	84.51
+LM	**PP-GLAM (Ours)**	**90.45**	**82.36**	**90.79**	**80.32**	**69.81**	**80.63**	**79.23**	**67.68**	**79.61**	**96.69**	**90.99**	**96.78**	**87.90**	**46.78**	**82.24**	**90.41**	**82.08**	**90.92**

baselines, we note that LM models tend to outperform the graph models, which underscores the importance of semantic features. Even in the case of LM models for the US region, we notice that the newer LM model (i.e., BigBird) outperforms the relatively older models such as DeBERTa and COCOLM, which strengthens our case for plug and play ensemble models with replaceable components. PP-GLAM, with its ensemble framework, improves the performance of both standalone LMs and GNNs by 4%-11% and 20%-75%, respectively. Moreover, the model can also leverage further research advances within LM models and GNNs in a non-intrusive manner. This enables us to take advantage of the latest works in an efficient pipeline. Additionally, we also see that PP-GLAM outperforms the SALAM model, which also utilizes both LM and GNN models. This is because SALAM gives more importance to graph features, which are not available during the evaluation phase. Thus, we conclude that an ensemble that can adapt its dependence on language and graph features according to the dataset is the best fit for a real-world industry scenario of search relevance.

Table 3: Comparison of memory and processing requirements for different models in the training and inference pipelines. The Columns indicate the number of parameters (Param), pre-training time (PTT), and fine-tuning time (FTT) in seconds per epoch, inference time (IT) in milliseconds per sample, VRAM requirement (Mem), and disk space requirement (Disk) in Gigabytes.

Model	Param	PTT	FTT	IT	Mem	Disk
LM	230M	128K	215	10.67	1.01	16.2
GNN	70M	133K	169	9.87	0.49	39.8
Relation-GNN	150M	130K	217	10.83	1.38	22.1
SALAM	279M	135K	231	11.05	2.57	169.1
PP-GLAM (Ours)	450M	138K	220	10.67	2.38	160.1
PP-GLAM (Red)	290M	-	-	10.50	0.98	160.1

4.3 RQ2: Practical Environment

To comprehend the suitability of PP-GLAM for industry settings, we compare its memory and processing requirements against the LM- and GNN-based baselines. We evaluate the models on the basis of their number of parameters (Param), pre-training time (PTT), fine-tuning time (FTT), inference time (IT), GPU RAM requirement (Mem), and disk space requirement (Disk).

Results shown in Table 3 demonstrate that our model's computational and memory requirements are comparable to that of the available alternatives and differ only by a negligible margin of 3%-5%. This margin is easily manageable due to the monthly update rate of query-product pairs, as discussed in Section 3.3. An important challenge, however, is the additional time taken by the Graph Extraction and De-noising product information modules. In our pipeline, we shift these modules to the pre-computation step and save the results as hash tables to enable constant-order retrieval. However, this leads to a lack of availability of certain graphs for the query-product pairs in the evaluation set and also adds to the disk space requirement, which increases proportionally to the number of unique query-product pairs. For the problem of search relevance, our model can inductively handle the graph unavailability problem by returning the results from the language model. Furthermore, the additional disk requirement is not a concern since it is relatively inexpensive. However, further applications of our model should consider these limitations.

4.4 RQ3: Feature Aggregation

To confirm the effectiveness of the GBDT ensemble as a feature aggregation mechanism, we compare its performance to other commonly used mechanisms of using multi-layer perceptrons (MLPs) layer and attention layers. Both the MLP and attention layer perform their feature aggregation in a latent space. This leads to a lack of interpretability and generalization over a new data distribution. In addition, both these approaches will require an expensive re-training step for the entire model. In contrast, our GBDT based ensemble consists of decision trees that are interpretable using additive SHAP values. Also, they are trained on the outputs obtained from different models, and thus, the training of the aggregation mechanism (ensemble) for a new data distribution does not require the re-training of component graph and language models. Furthermore, from Table 4, we note that GBDT ensembles perform better

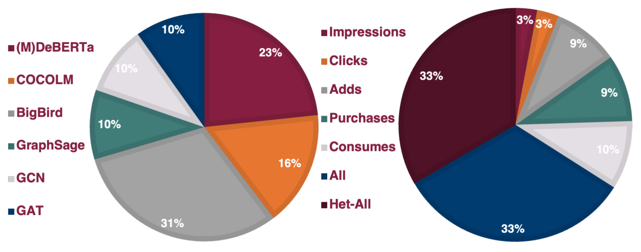
Table 4: Comparison of different feature aggregation methods (MLP and attention) and the constrained model variant (PP-GLAM (Red)) of our model for the search relevance tasks of ESCI classification and irrelevant classification. Evaluation metrics presented in the columns are accuracy (Acc), macro-F1 (MacF1), and weighted F1 (WtF1). The best results are in bold font.

Task	ESCI Classification									Irrelevant Classification								
Locale	United States (US)			Spain (ES)			Japan (JP)			United States (US)			Spain (ES)			Japan (JP)		
Model	Acc	MacF1	WtF1	Acc	MacF1	WtF1	Acc	MacF1	WtF1	Acc	MacF1	WtF1	Acc	MacF1	WtF1	Acc	MacF1	WtF1
MLP	81.18	74.06	79.92	69.64	60.76	71.94	70.41	56.98	69.15	86.78	81.82	85.19	76.21	40.72	73.38	80.35	69.10	78.97
Attention	85.20	76.77	85.46	75.22	64.05	75.18	74.04	62.09	74.47	91.08	84.81	91.10	82.32	42.92	76.68	84.49	75.30	85.05
PP-GLAM (Ours)	**90.45**	**82.36**	**90.79**	**80.32**	**69.81**	**80.63**	**79.23**	**67.68**	**79.61**	**96.69**	**90.99**	**96.78**	**87.90**	**46.78**	**82.24**	**90.41**	**82.08**	**90.92**
PP-GLAM (Red)	89.04	80.86	89.31	79.03	68.43	79.50	78.07	66.28	78.36	95.18	89.33	95.20	86.49	45.86	81.09	89.09	80.38	89.49

by 6%-9% at feature aggregation than their alternatives because they can better generalize over the evaluation dataset.

4.5 RQ4: Additive Explanations

In this experiment, we study the interpretability of our model by analyzing the contribution of LM and GNN models and different behavioral signals using additive SHAP explanations and aim to compress the model for inference by removing the models with lower contribution.



(a) Model contribution (b) Relation contribution

Figure 4: SHAP value-based additive contribution of different features to the overall performance of our model. (a) provides the contribution of different models, and (b) shows the contribution of different relations.

From Figure 4a, we observe that LM models (around 16%-31%) contribute significantly higher to our model's performance compared to GNN models (at 10% each). Diving deeper into the contribution of different relations, we notice that GNN models that use all features either as homogeneous graphs or heterogeneous graphs have the highest contribution at 33%, followed by strong behavioral signals such as adds and purchases at 9%-10%. Finally, relatively weak behavioral signals have an insignificant contribution of 3% to the model performance. Based on these contributions, we construct an ensemble of reduced size, PP-GLAM (Red), with the highest contribution models of (M)DeBERTa, BigBird, and GraphSage with Het-All relations. We observe that the reduced ensemble leads to a 57.8% reduction in the number of parameters (Table 3) with a performance reduction of only 1.6% (Table 4).

5 DISCUSSION

This section describes a strategy to deploy PP-GLAM in an industrial setting and then understand its impact on the search engine.

Deployment Strategy: As explained in Section 3.3, PP-GLAM can be trained and ensembled offline using annotated labels. The trained models can then be stored as checkpoints for inference. These checkpoint models will be loaded in parallel GPUs using LuaJIT platform, which enables faster real-time inference by removing the overhead of loading weights for each iteration. During inference, the model processes query-product pairs in batches and saves the predicted labels for further utilization in downstream tasks. The integration of the PP-GLAM method into existing workflows that utilize query-product pairs as input for classification is a simple task with minimal modifications required. The graph preprocessing step is also implemented with efficient caching and real-time lookup mechanism with constant order retrieval. In particular, given the long-tailed distribution of product search queries, we can pre-compute query and product neighborhoods for the most frequently seen samples. Additionally, efficient online inference methods [14] are employed to compute representations for less frequently seen queries in real time. The inclusion of graph information results in significant performance gains, while the impact on inference time is negligible. Thus, the deployment of PP-GLAM is a feasible and viable investment for practical applications.

Impact on Search Engine: In this section, we present the impact of incorporating an ESCI classifier in a product search engine. The ranking of e-commerce products relies on lexical, behavioral, and semantic matching. However, behavioral data can be unreliable, leading to biases in the data used to train these models. By implementing an ESCI-based classification at the final stage, or using these classification predictions as inputs for downstream models, these issues can be mitigated. Additionally, the results from these models can be utilized for customer messaging, providing explanations for why a particular item was shown in response to a query. Improving the ESCI classifiers' performance can both enhance the quality of product search and the customers' trust in these systems, ultimately allowing practitioners to better serve their end-users.

6 CONCLUSION

In this paper, we presented PP-GLAM, a modular ensemble of LMs and relational GNNs that utilizes GBDTs for flexible model selection in a practical environment. Through our experimental evaluation on a multi-regional public e-commerce dataset, we have shown the effectiveness of PP-GLAM on search relevance and irrelevant detection tasks of ESCI classification, outperforming current alternatives. We have also demonstrated the compatibility of PP-GLAM with industry settings and highlighted the benefits of using ensemble methods as an interpretable strategy to aggregate semantic and behavioral signals and efficiently select the most impactful models. Additionally, we have detailed a deployment strategy for integrating our framework in practical settings with dynamic data sources. Overall, our approach presents a promising step towards improved e-commerce product search.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations*, Yoshua Bengio and Yann LeCun (Eds.).

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems*, C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (Eds.), Vol. 26. Curran Associates, Inc.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901.

[4] Nurendra Choudhary, Charu C. Aggarwal, Karthik Subbian, and Chandan K. Reddy. 2022. Self-Supervised Short-Text Modeling through Auxiliary Context Generation. *ACM Trans. Intell. Syst. Technol.* 13, 3, Article 51 (apr 2022), 21 pages. https://doi.org/10.1145/3511712

[5] Nurendra Choudhary, Nikhil Rao, Karthik Subbian, and Chandan K. Reddy. 2022. Graph-Based Multilingual Language Model: Leveraging Product Relations for Search Relevance. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 2789–2799. https://doi.org/10.1145/3534678.3539158

[6] Nurendra Choudhary and Chandan K Reddy. 2023. Complex Logical Reasoning over Knowledge Graphs using Large Language Models. *arXiv preprint arXiv:2305.01157* (2023).

[7] Alexis Conneau and Guillaume Lample. 2019. *Cross-Lingual Language Model Pretraining.* Curran Associates Inc., Red Hook, NY, USA.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[9] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 855–864. https://doi.org/10.1145/2939672.2939754

[10] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 1025–1035.

[11] Junheng Hao, Tong Zhao, Jin Li, Xin Luna Dong, Christos Faloutsos, Yizhou Sun, and Wei Wang. 2020. P-Companion: A Principled Framework for Diversified Complementary Product Recommendation. 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, 2517–2524. https://doi.org/10.1145/3340531.3412732

[12] Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing. https://doi.org/10.48550/ARXIV.2111.09543

[13] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. In *International Conference on Learning Representations.*

[14] Ziheng Jiang, Tianqi Chen, and Mu Li. 2018. Efficient deep learning inference on edge devices. In *SysML 2018.*

[15] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc.

[16] Simran Khanuja, Melvin Johnson, and Partha Talukdar. 2021. MergeDistill: Merging Language Models using Pre-trained Distillation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Association for Computational Linguistics, Online, 2874–2887. https://doi.org/10.18653/v1/2021.findings-acl.254

[17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

[18] V. Klema and A. Laub. 1980. The singular value decomposition: Its computation and some applications. *IEEE Trans. Automat. Control* 25, 2 (1980), 164–176. https://doi.org/10.1109/TAC.1980.1102314

[19] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations.*

[20] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 4768–4777.

[21] Mohammad Najah Mahdi, Abdul Rahim Ahmad, Qais Saif Qassim, Mohammed Ahmed Subhi, and Taofiq Adeola Bakare. 2022. A Survey on the Use of Personalized Model-Based Search Engine. In *Proceedings of International Conference on Emerging Technologies and Intelligent Systems*, Mostafa Al-Emran, Mohammed A. Al-Sharafi, Mohammed N. Al-Kabi, and Khaled Shaalan (Eds.). Springer International Publishing, Cham, 88–98.

[22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2* (Lake Tahoe, Nevada) *(NIPS'13)*. Curran Associates Inc., Red Hook, NY, USA, 3111–3119.

[23] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. 2017. graph2vec: Learning Distributed Representations of Graphs. In *Proceedings of the 13th International Workshop on Mining and Learning with Graphs (MLG)*.

[24] Dai Quoc Nguyen, Tu Dinh Nguyen, and Dinh Phung. 2022. Universal Graph Transformer Self-Attention Networks. In *Companion Proceedings of the Web Conference 2022* (Virtual Event, Lyon, France) *(WWW '22)*. Association for Computing Machinery, New York, NY, USA, 193–196. https://doi.org/10.1145/3487553.3524258

[25] Priyanka Nigam, Yiwei Song, Vijai Mohan, Vihan Lakshman, Weitian (Allen) Ding, Ankit Shingavi, Choon Hui Teo, Hao Gu, and Bing Yin. 2019. Semantic Product Search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 2876–2885. https://doi.org/10.1145/3292500.3330759

[26] Manuel Duarte Ortigueira and Miguel-Angel Lagunas. 1991. Eigendecomposition versus singular value decomposition in adaptive array signal processing. *Signal Processing* 25, 1 (1991), 35–49. https://doi.org/10.1016/0165-1684(91)90037-J

[27] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric Transitivity Preserving Graph Embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 1105–1114. https://doi.org/10.1145/2939672.2939751

[28] S.K. Pal and S. Mitra. 1992. Multilayer perceptron, fuzzy sets, and classification. *IEEE Transactions on Neural Networks* 3, 5 (1992), 683–697. https://doi.org/10.1109/72.159058

[29] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1532–1543. https://doi.org/10.3115/v1/D14-1162

[30] Chandan K. Reddy, Lluís Màrquez, Fran Valero, Nikhil Rao, Hugo Zaragoza, Sambaran Bandyopadhyay, Arnab Biswas, Anlu Xing, and Karthik Subbian. 2022. Shopping Queries Dataset: A Large-Scale ESCI Benchmark for Improving Product Search. (2022). arXiv:2206.06588

[31] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text mining: applications and theory* (2010), 1–20.

[32] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. In *ICML 2020 Workshop on Graph Representation Learning.*

[33] Sam T. Roweis and Lawrence K. Saul. 2000. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science* 290, 5500 (2000), 2323–2326. https://doi.org/10.1126/science.290.5500.2323

[34] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE transactions on neural networks* 20, 1 (2008), 61–80.

[35] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning Semantic Representations Using Convolutional Neural Networks for Web Search. In *Proceedings of the 23rd International Conference on World Wide Web* (Seoul, Korea) *(WWW '14 Companion)*. Association for Computing Machinery, New York, NY, USA, 373–374. https://doi.org/10.1145/2567948.2577348

[36] Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. 2021. Fast WordPiece Tokenization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 2089–2103. https://doi.org/10.18653/v1/2021.emnlp-main.160

[37] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems* 27 (2014).

[38] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-Scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web* (Florence, Italy) *(WWW '15)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1067–1077. https://doi.org/10.1145/2736277.2741093

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc.

[40] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations* (2018).

[41] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous Graph Attention Network. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 2022–2032. https://doi.org/10.1145/3308558.3313562

[42] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc.

[43] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do Transformers Really Perform Badly for Graph Representation?. In *Thirty-Fifth Conference on Neural Information Processing Systems.*

[44] Changlong Yu, Hongming Zhang, Yangqiu Song, and Wilfred Ng. 2022. CoCoLM: Complex Commonsense Enhanced Language Model with Discourse Relations. In *Findings of the Association for Computational Linguistics: ACL 2022.* Association for Computational Linguistics, Dublin, Ireland, 1175–1187. https://doi.org/10.18653/v1/2022.findings-acl.93

[45] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems* 33 (2020), 17283–17297.

[46] Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. 2022. GreaseLM: Graph REASoning Enhanced Language Models. In *International Conference on Learning Representations.*

A ALGORITHMS

Algorithms 1 and 3 provide the algorithm for our model's selection pipeline, and our model's inference pipeline, respectively. Algorithm 2 presents the model's training flow.

Algorithm 1: Model Selection.

```
Input: Current model set $M$, candidate models $LM, GNN$,
       inference metric $\Lambda$, constraint $K$;
Output: New model set $M$;
1  if $M = \emptyset$ then
2      # Sort candidate models based on SHAP values
3      $R = sort_{\Omega_f}(\{LM \cup GNN\})$
4      Initialize $i = 1, \Lambda = 0$;
5  else
6      # Sort candidate and $|LM \cup GNN|$ least performing models in $M$
7      $lst(M) = \arg\min_{\Omega, |LM \cup GNN|}(M)$
8      $R = sort_{\Omega_f}(\{LM \cup GNN \cup lst(M)\})$
9      Initialize $i = 1, \Lambda = \sum_{m \in M \setminus lst(M)} \Lambda_m$;
10 end
11 while $\Lambda \leq K$ & $i \leq |R|$ do
12     $\Lambda = \Lambda + \Lambda_m$;
13     $M = M \cup R[i]$;
14 end
15 return $M$
```

Algorithm 2: PP-GLAM training flow.

```
Input: Query-product pairs $(Q, P) = \{(q, p)\}$,
       Pre-computed neighborhoods $\mathcal{G}^{\psi} = \{\mathcal{G}^{\psi}(q, p)\}$,
       Ground truth $y$;
Output: Predictor $P_\theta, \theta$;
1  Initialize model parameters $\theta$;
2  for number of epochs; until convergence do
3      Initialize loss $l = 0$;
4      for $(q, p) \in (Q, P), \mathcal{G}^{\psi}(q, p) \in \mathcal{G}^{\psi}$ do
5          Tokenize input $t = q \| [SEP] \| p'$;
6          # Process through language models
7          for $LM_i \in LM$ do
8              $\hat{y}_i^{LM}(t) = \phi_d^l(LM_i(t))$; via Eq. (3)
9          end
10         $\hat{Y}_{LM}(t) = \left\{\hat{y}_i^{LM}(t) \middle| i \in [1, |LM|]\right\}$; via Eq. (4)
11         # Aggregate node neighborhoods
12         for $GNN_i \in GNN$ do
13             $h_{k+1} = \sigma\left(D^{\zeta-1}\mathcal{G}^{\psi}(q, p)D^{\zeta}h_k W_k\right)$
14             $\hat{y}_i^{GNN}(h_0) = \phi_d^l(h_{k+1}[i_q] \| h_{k+1}[i_p])$; via Eq. (6)
15         end
16         $\hat{Y}_{GNN}(h_0) = \left\{\hat{y}_i^{GNN}(h_0) \middle| i \in [1, |GNN|]\right\}$; via Eq. (7)
17         # GBDT Ensemble
18         $\hat{Y} = \hat{Y}_{LM}(t) \| \hat{Y}_{GNN}(h_0) \| f$
19         $\hat{y} = \frac{1}{T}\sum_{t=1}^{T} o_t(\hat{Y})$; via Eq. (8)
20         # Loss Calculation
21         $L(y, \hat{y}) = -\sum_{y \in Y}(ylog(\hat{y}) + (1 - y)log(1 - \hat{y}))$; via Eq. (9)
22         $l = l + L(y, \hat{y})$;
23     end
24     # Update parameters
25     $\theta = \theta - \alpha \nabla_\theta L(y, \hat{y})$; via Eq. (10)
26 end
27 return $P_\theta, \theta$
```

Algorithm 3: PP-GLAM inference flow.

```
Input: $\{(q, p)\}$, $\{\mathcal{G}^{\psi}(q, p)\}$, Model set $M$;
Output: Label $\hat{y}$;
1  $t = q \| [SEP] \| p'$;
2  for $LM_i \in LM_M$ do
3      $\hat{y}_i^{LM}(t) = \phi_d^l(LM_i(t))$; via Eq. (3)
4  end
5  $\hat{Y}_{LM}(t) = \left\{\hat{y}_i^{LM}(t) \middle| i \in [1, |LM|]\right\}$; via Eq. (4)
6  for $GNN_i \in GNN_M$ do
7      $h_{k+1} = \sigma\left(D^{\zeta-1}\mathcal{G}^{\psi}(q, p)D^{\zeta}h_k W_k\right)$
8      $\hat{y}_i^{GNN}(h_0) = \phi_d^l(h_{k+1}[i_q] \| h_{k+1}[i_p])$; via Eq. (6)
9  end
10 $\hat{Y}_{GNN}(h_0) = \left\{\hat{y}_i^{GNN}(h_0) \middle| i \in [1, |GNN|]\right\}$; via Eq. (7)
11 $\hat{Y} = \hat{Y}_{LM}(t) \| \hat{Y}_{GNN}(h_0) \| f$
12 $\hat{y} = \frac{1}{T}\sum_{t=1}^{T} o_t(\hat{Y})$; via Eq. (8)
13 return $\hat{y}$
```