Graph-based Multilingual Language Model: Leveraging Product Relations for Search Relevance

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

Nikhil Rao
Amazon, Palo Alto, USA
nikhilsr@amazon.com

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

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

ABSTRACT

The large-scale nature of product catalog and the changing demands of customer queries makes product search a challenging problem. The customer queries are ambiguous and implicit. They may be looking for an exact match of their query, or a functional equivalent (i.e., substitute), or an accessory to go with it (i.e., complement). It is important to distinguish these three categories from merely classifying an item for a customer query as relevant or not. This information can help direct the customer and improve search applications to understand the customer mission. In this paper, we formulate search relevance as a multi-class classification problem and propose a graph-based solution to classify a given query-item pair as exact, substitute, complement, or irrelevant (ESCI). The customer engagement (clicks, add-to-cart, and purchases) between query and items serve as a crucial information for this problem. However, existing approaches rely purely on the textual information (such as BERT) and do not sufficiently focus on the structural relationships. Another challenge in including the structural information is the sparsity of such data in some regions. We propose Structure-Aware multilingual LAnguage Model (SALAM), that utilizes a language model along with a graph neural network, to extract region-specific semantics as well as relational information for the classification of query-product pairs. Our model is first pre-trained on a large region-agnostic dataset and behavioral graph data and then fine-tuned on region-specific versions to address the sparsity. We show in our experiments that SALAM significantly outperforms the current matching frameworks on the ESCI classification task in several regions. We also demonstrate the effectiveness of using a two-phased training setup (i.e., pre-training and fine-tuning) in capturing region-specific information. Also, we provide various challenges and solutions for using the model in an industrial setting and outline its contribution to the e-commerce engine.

CCS CONCEPTS

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

KEYWORDS

Graphs, language models, search relevance, e-commerce

*Also with Amazon, Palo Alto, USA.

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08.
https://doi.org/10.1145/3534678.3539158

ACM Reference Format:
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 (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539158

1 INTRODUCTION

The goal of an e-commerce search engine is to show a ranked list of items that best match a shopper's query intent. Unlike standard search engines [13, 17], e-commerce engines cannot solely rely on the textual and semantic content of a query. This is because shoppers' queries are often ambiguous, broad, lack specific intent or have an implicit intent. A simple one-size-fits-all approach to e-commerce product retrieval seldom works for all types of customer queries. Practitioners often mitigate this using additional human labeled data to re-rank items or aptly message the shoppers as to why an item was surfaced. The shopper may be looking for an *exact* match for their query, or a functional equivalent (i.e., *substitute*), or an accessory to go with it (i.e., *complement*). It is important to distinguish these three categories from merely classifying an item as relevant or not for a given query. Specifically, annotated data is used to classify query and item pairs into Exact (E), Substitute (S), Complement (C), and Irrelevant (I) (ESCI) classes.

Obtaining human annotated data is time consuming and costly. The size of datasets available to train ESCI classification models is orders of magnitude smaller than those used to train relevance models [4] (which typically rely on anonymized aggregated customer shopping behavior). In addition, the labeled data available per-region to train these models is even smaller. However, there is additional information that is present in e-commerce datasets, in the form of query-item graphs capturing interactions between these pairs, and co-purchase behavior across products in the catalog. We hypothesize that one can train better ESCI classifiers by aggregating information along nodes and edges of this graph to extract queries and items that are similar to the ones in the labeled dataset.

Models that use graph information have shown to be competitive in e-commerce search applications. These models typically tend to outperform their pointwise or pairwise counterparts such as C-DSSM [21, 28] and MV-LSTM [32], and their adversarial [20] or multilingual [1] extensions. This is because the aforementioned methods only focus on the textual information in the query-product pairs, and ignore any relational information across queries and products. However, existing approaches work with very small graphs, and are not built to scale to the sizes of graphs typically seen in the real-world applications. In this paper, we propose **Structure-Aware**

multilingual LAnguage Model (SALAM), illustrated in Figure 2, that combines text and graph information in a scalable manner, overcoming the drawbacks of the aforementioned methods.

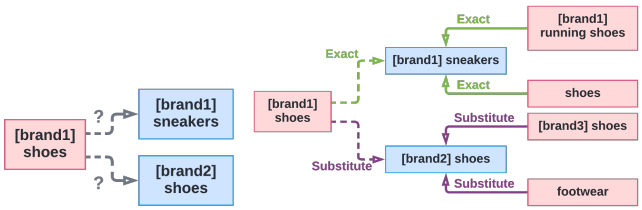


Figure 1: Sample of a query-product pair with (a) only text and (b) text and graph structure. Note that classifying the relation in (a) based only on text is difficult because both the products have a single token match with the query. However, with the aid of graph structure in (b), the classification becomes easier with more semantic token matches and additional information about the relations.

Academic research in this topic has explored certain solutions to the problem of integrating text and graph using hybrid techniques such as TextGNN [40], TextGCN [35], and Graphormer [37]. However, these approaches are not scalable to industrial applications which typically need to handle hundreds of millions of queries everyday and require low processing latency.

SALAM is first pretrained on a corpus of labeled data across multiple regions as well as the query-product graph. This allows for a model that can be initialized using *all* the labeled data available. Next, SALAM is fine-tuned on region-specific labeled datasets. We show via extensive experiments that the use of auxiliary graph structured information and the proposed two-stage training approach vastly outperforms baseline methods by 10%-65% across multiple regions. The specific architectural choices in SALAM also allow it to be deployed in web-scale systems. SALAM utilizes a multilingual language model (MLM) [5] in congruence with a Graph Convolution Network (GCN) [15], to capture the textual and graph features of query-product pairs, respectively. These features are combined in the latent space with an attention layer [30] to capture the inter-dependency between the textual and graph-based features, which are further utilized for the final ESCI classification. To summarize, the major contributions of this paper are as follows:

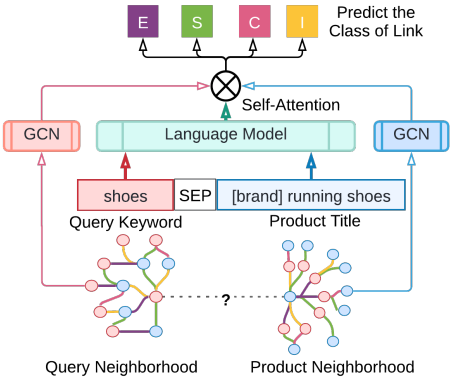


Figure 2: An overview of the proposed SALAM. Given the input query-product pair with their corresponding local neighborhoods, SALAM combines the power of textual and graph representations for ESCI classification.

(1) We present Structure-Aware multilingual LAnguage Model (SALAM) that combines a MLM and GCNs within an attention-based framework to aggregate features from both the text and graph attributes of query-product pairs, respectively. We show that the proposed unique aggregation mechanism is able to outperform the current state-of-the-art models on the task of ESCI classification with similar computational resources and model latency limits.
(2) We demonstrate the benefits of utilizing the graph features and attention aggregation through an ablation study.
(3) We illustrate the performance gains in multi-region/multilingual classification produced by using a two-phase (pre-training and fine-tuning) approach over one that trains a single model.
(4) We detail the challenges that affect the practical viability of SALAM in industrial settings, and provide solutions for the same. We also outline the primary benefits of our approach to an e-commerce search engine.

The rest of the paper is organized as follows; Section 2 presents the related literature in this topic. Section 3 defines our problem statement and the model architecture. Section 4 describes our experimental results in a comparative evaluation and Section 5 presents our deployment strategy along with the contributions of the model to the e-commerce engine. Section 6 concludes the paper.

2 RELATED WORK

In this section, we describe the earlier works relevant to our current problem. We present three broad areas of representation learning: graph, text, and hybrid (graph+text) methods.

Graph Representation Learning. Earlier research into graph embeddings relied on two broad methods: matrix factorization and random walks. Matrix factorization methods [22, 26] attempt to factorize the adjacency matrix A to a low-dimensional representational matrix L by minimizing $\|L^T L - A\|$. Random walk techniques [10, 19, 25, 29, 33] utilize a traversal along the edge connections of a node to retrieve its neighborhood and learn the representation. These approaches form a vector space model based on the node's neighborhood and have shown the importance of learning graph representations. However, these methods depend on the methodology of the random walk traversal technique and do not handle the entirety of a node's neighborhood. Hence, we utilize the power of deep networks to aggregate features from the nodes' neighborhood. Some of the popular deep architectures include Graph Neural Network (GNN) [27], GraphSage [11], Graph Convolution Network (GCN) [15], and Graph Attention Network (GAT) [31]. In our proposed SALAM model, we utilize the scalable GCN architecture for learning features from the graph structure of query-product pairs.

Text Representation Learning. Early research in text embeddings relied on capturing the context of individual words (or tokens). These approaches [18, 24] show the effectiveness of distributional representations in extracting semantic features. However, most of the problems that involve text need sentence embeddings and not word-level representations. Towards this, text representations evolved to utilize sequential word information using recurrent networks [32], convolution networks [28], and attention networks [2].

Recently, newer techniques are based on learning a pre-trained language model [6, 8] that can be fine-tuned on specific tasks. Such approaches have shown improved performance over the other representational techniques. Thus, in our model, we utilize this two-step approach, i.e., pre-training and fine-tuning steps, to capture region-agnostic and region-specific information, respectively.

Hybrid (Graph + Text) Representation Learning. Recently, we see an increasing research effort focused on extracting hybrid graph and text information from structured datasets. TextGNN [40] and TextGCN [35] utilize text embeddings to initialize node features in a graph network framework. Graphormers [38] utilize manual graph features (such as spatial encoding and centrality measures) within a text-based transformer architecture to capture hybrid features. These methods, while effective, are not scalable to a practical application that receives millions of queries per day and hence, we developed SALAM as an effective architecture with low latency. Another line of approaches [9, 36] focus on the problem of structured reasoning for question answering. However, these approaches require specific relational knowledge graphs which are, generally, not readily available and require significant investment for construction. Our model uses the platform generated purchase information for knowledge and hence, does not need any additional investment.

3 MODEL ARCHITECTURE

In this section, we first discuss the problem statement and then explain the different components of our architecture and detail its training and inference pipeline.

3.1 Problem Statement

Let us say that the input query-product pair is (Q, P). From a global graph $\mathcal{G}$, we pre-compute their neighborhood based on historical purchase information as $Q_{\mathcal{G}}$ and $P_{\mathcal{G}}$, respectively. The goal of this work is to build a prediction model P_θ with parameters θ such that

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

where $\hat{y}$ is the predicted output of model P_θ for a corresponding input sample $x = (Q, P, Q_{\mathcal{G}}, P_{\mathcal{G}})$ and $y \in \{E, S, C, I\}$ is the ground truth label for the sample x.

3.2 Model Components

In this section, we describe the function of different components of SALAM and show their hybrid graph-text feature extraction and aggregation methods. As shown in Figure 3a, we begin the pipeline with an input query-product pair either received from a user or a collection of historical queries, $x = (Q, P)$. Using x, we compute the corresponding graph neighborhoods $x_{\mathcal{G}} = (Q_{\mathcal{G}}, P_{\mathcal{G}})$ using the Graph Construction module. The textual features of the input pair are then encoded using a language model and the neighborhood $x_{\mathcal{G}}$ is aggregated using a Neighborhood aggregation network to produce two complementary representations $t_{Q,P}$ (based on text) and $g_{Q,P}$ (based on graph). These representations are then pooled together using attention to finally produce the probabilities $P_\theta(y'|x,\theta)$ using a normalizing function (i.e., softmax). We will now provide more specific details of these components used in our model.

Graph Construction. In this module, we pre-compute a local neighborhood of connections between queries and products. Several such query-product graphs are, generally, available in the context of e-commerce engines such as purchase information, retrieval history, and click-through rates. For simplicity, in our problem, we utilize the purchase information, i.e., a query node and product node have an edge between them if the product has been purchased for a given query, else no edge exists. To construct the neighborhood, we perform a breadth-first traversal over the root node till a depth threshold of n[1]. Additionally, we also store the corresponding nodes' features as semantic encoding of its text, obtained from a pre-trained XLM model [6]. The graph $x_{\mathcal{G}} = (Q_{\mathcal{G}}, P_{\mathcal{G}})$ for query-product pair $x = (Q, P)$ is stored in a hash map for constant time retrieval during the training and inference phase.

Language Model. To encode the text attributes (Q, P), we adopt the multilingual XLM [6] model[2]. For a pair of text sequences, the language model provides a sequence of encoding of $e \in \mathbb{R}^F$, which is further reduced to $t \in \mathbb{R}^f$, (where $f \ll F$) through a dense linear layer, $\phi_F^f : \mathbb{R}^F \rightarrow \mathbb{R}^f$ for computational efficiency in the attention pooling and final classification layer.

$$t_{Q,P} = \left\{\phi_F^f(XLM(Q,P))\right\}^f \tag{2}$$

where $\{X\}^f$ implies that X is a feature vector of f dimensions.

Neighborhood Aggregation. In SALAM, we use a Graph Convolution (GCN) encoder [15] to encode the graph neighborhood of the query-product pair. The choice of the graph encoder is dependent on the trade-off between its performance and scalability on the huge e-commerce purchase information. From our experiments, described in Section 4.4, we chose GCN due to its performance advantage over other graph encoders with similar parameters. Let us say the input to a GCN for a neighborhood of N nodes with feature dimensions F is given by $h_0 \in \mathbb{R}^{N \times F}$. An L-layer GCN aggregates the node neighborhood into features as:

$$h_{l+1} = \sigma\left(D^{-\frac{1}{2}} A D^{-\frac{1}{2}} h_l W_l\right) \tag{3}$$

$$\text{Thus, } GCN(h_0) = \{h_L\}^f \tag{4}$$

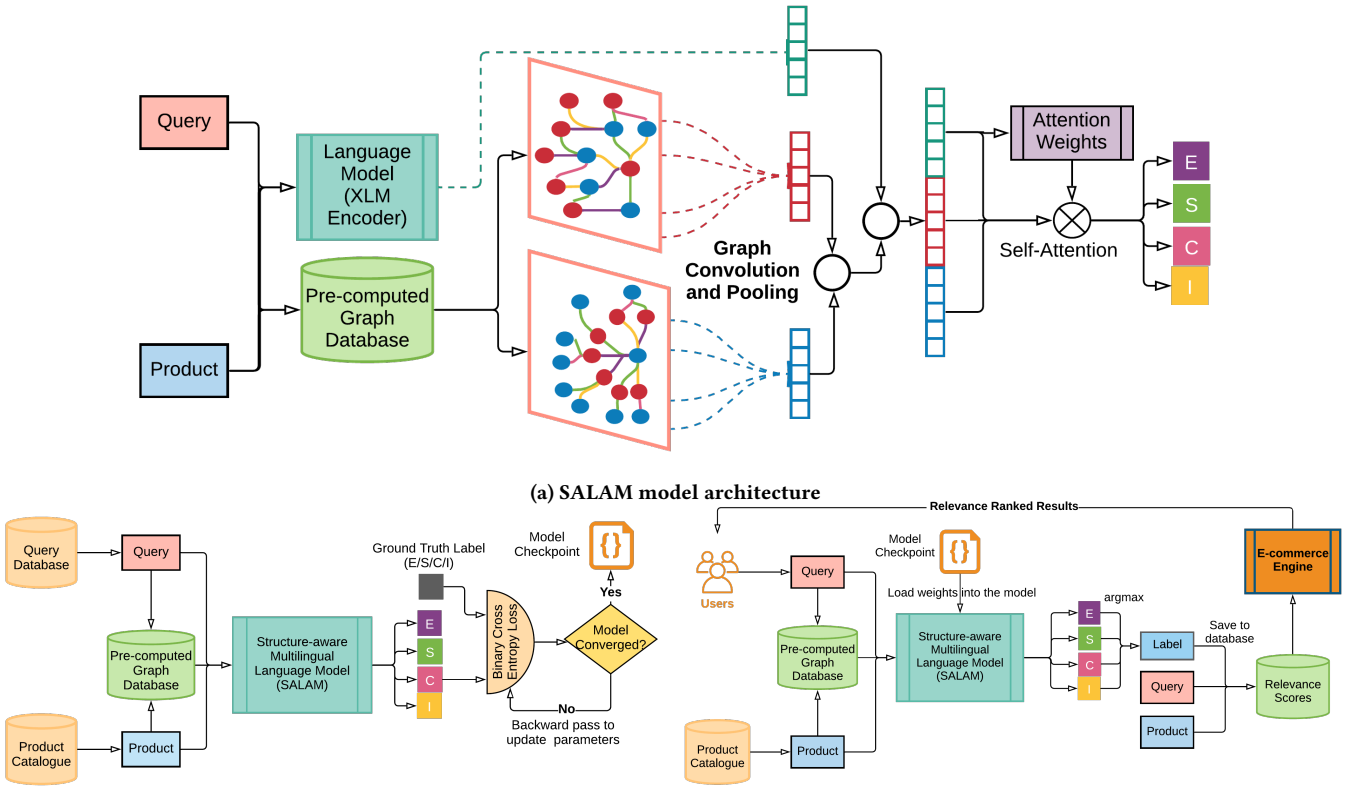
where h_l and h_{l+1} are the input and output to the l^{th} GCN layer. W_l is the learnable feature map for the layer and σ is a non-linear activation function. $D \in \mathbb{R}^{N \times N}$ and $A \in \mathbb{R}^{N \times N}$ are the diagonal degree and adjacency matrices of the graph, respectively. h_L is the output of the final GCN layer and consequently the encoding of the nodes' neighborhood.

Applying this to the given input graph of query-product pair $x_{\mathcal{G}} = (Q_{\mathcal{G}} \in \mathbb{R}^{N \times F}, P_{\mathcal{G}} \in \mathbb{R}^{N \times F})$ with node features $Q, P \in \mathbb{R}^F$, we independently encode the neighborhood graphs of both the query and product using GCNs and then concatenate ($\circ$) the resultant features for the final output, $g_{Q,P}$ as given below:

$$\{g_{Q,P}\}^{2f} = \{GCN(Q_{\mathcal{G}})\}^f \circ \{GCN(P_{\mathcal{G}})\}^f \tag{5}$$

[1] Due to computational constraints, we empirically set n to be 2.

[2] XLM is chosen due to its popularity and multilingual nature, any other trainable multilingual text encoder can be used as an alternative.



(a) SALAM model architecture

(b) Training pipeline

(c) Inference pipeline

Figure 3: An overview of SALAM's architecture (a) and its training (b) and inference (c) pipelines. (a) The inputs to the architecture are a query-product text pair with a pre-computed graph neighborhood which is obtained from the purchase information. The textual and graph features are encoded using a language model and GCN, respectively. Subsequently, the concatenation of the features are pooled with an attention mechanism to finally predict the probability of the classes. (b) The training pipeline uses the query database and product catalogue with a ground truth label for back-propagation and saves a model checkpoint on convergence. (c) The inference pipeline employs the checkpoint to classify search relevance on a user query and products from the catalogue to generate the final label, which is stored in the database for relevance computation in the e-commerce engine.

Attention Pooling. The graph and text features, while encoded independently, are derivatives of the same e-commerce dataset. Hence, our current encoding misses the inter-relations between the features. To capture these dependencies, we rely on an attentive pooling mechanism [30] that calculates attention weights to higher order interactions between the nodes' text and graph features to subsequently combine them into a single feature set. The final output of the attentive pooling between the graph and text features, $o_{Q,P}$, is computed as:

$$\{e_{Q,P}\}^{3f} = \{t_{Q,P}\}^{f} \circ \{g_{Q,P}\}^{2f} \tag{6}$$

$$\{q_{Q,P}\}^{h} = \phi^{h}_{3f}(e_{Q,P}) \; ; \; \{k_{Q,P}\}^{h} = \phi^{h}_{3f}(e_{Q,P}) \; ; \; \{v_{Q,P}\}^{h} = \phi^{h}_{3f}(e_{Q,P}) \tag{7}$$

$$\{a_{Q,P}\}^{h} = \left\{ \frac{exp(\alpha_{ij})}{\sum_{j=0}^{h} exp(\alpha_{ij})} v_i \right\}_{i=0}^{h} \; ; \quad \{\alpha_{ij}\}^{1} = \frac{q_i k_j}{\sqrt{2h}} \tag{8}$$

$$\{o_{Q,P}\}^{|y|} = Softmax\left(\phi^{|y|}_{h}(a_{Q,P})\right) \tag{9}$$

where h is the number of hidden units in the attention layers and, $q_{Q,P}$, $k_{Q,P}$ and $v_{Q,P}$ are the query, key and value vectors, respectively, obtained using independent linear transformation layers $\phi^{h}_{3f} : \mathbb{R}^{3f} \rightarrow \mathbb{R}^{h}$ for the attention layer. α_{ij} is the attention weight assigned to the interaction between i^{th} query and j^{th} key feature. $\phi^{|y|}_{h} : \mathbb{R}^{h} \rightarrow \mathbb{R}^{|y|}$ is a dense layer for feature compression to the final number of classes and $Softmax(\cdot)$ is used to map the final set of features to the probability space for loss computation as:

$$L(o_{Q,P}, y_k) = -\sum_{k=1}^{|y|} y_k \log(o_{Q,P}); \; y_k = \begin{cases} 1, \text{if class is k} \\ 0, \text{otherwise} \end{cases} \tag{10}$$

Handling Multi-Regionality. SALAM uses a two-phase approach of pre-training on the all the regions together and further fine-tuning on particular regions to utilize the entire volume of training data and add regional specificity, respectively. This approach shows significant improvement in performance (as shown in Section 4.3) compared to the single phase training framework.

3.3 Implementation Details

SALAM is implemented using Pytorch framework [23] on a single Nvidia V100 GPU with 4GB VRAM. The model is optimized using AdamW [16] with standard beta parameters of 0.9 and 0.999 and a weight decay rate of 0.01. The language model adopted for text

encoding is XLM-Roberta-base with the standard parameter setting given in [5]. We limit our node neighborhood to a depth of $n = 2$ with a maximum neighborhood size of $N = 100$. For the graph encoding, we utilize a 2-layer GCN model ($L = 2$) with 16 hidden units. The raw feature set of the nodes and text encoding of the language model is of $F = 768$ dimensions which is scaled down to $f = 16$ (empirically determined) for computational efficiency. The pooling is done on a single head attention network with $h = 8$ hidden units in the linear transformations. The loss function used for gradient descent is binary cross entropy weighted by the class distribution of the samples in the training dataset. SALAM is trained and evaluated on three classification variants of the ESCI search relevance with following unique downstream applications: (i) E vs S vs C vs I, (ii) E vs S vs CI, and (iii) E vs SCI. The code and evaluation of our method is already made publicly available at https://github.com/amazon-research/structure-aware-language-models/.

3.4 Training and Inference Pipeline

In an industrial setting, the training and inference phases have their unique set of requirements both in terms of model latency and complexity. In this section, we explain our pipelines that were designed to meet the general industry requirements.

During the **training phase**, depicted in Figure 3b, the model can have access to a significant amount of both memory (RAM/disk) and compute power (GPUs). More importantly, we also observe that a significant portion (70%-80%) of queries received by e-commerce search engines remain constant over the days of a month, and thus, we consider a monthly update cycle for our model checkpoints. Hence, the training phase also gets a significant amount of time due to the monthly entropy of the underlying dataset (i.e., update frequency). Thus, SALAM's dependence on graph pre-computation (a time intensive operation) and query-product pairing (a memory intensive operation) can be managed in practice. However, one limitation remains that the model checkpoints need to be industry-compliant, and hence, the number of model parameters are resource-dependent (generally a single GPU). For this, we choose our hyper-parameters (as given in Section 3.3) to limit model size to 2.5 GB (remaining GPU memory post overhead allocation). The training pipeline receives the query-product pair from the query database and product catalogue, with a manually assigned ESCI label. This information is used to pre-compute the node neighborhoods from purchase information and to train the SALAM architecture, whose parameters are further stored as a checkpoint upon convergence. The checkpoint is further finetuned to region-specific checkpoints using the same procedure, but with only region-specific datasets.

For the **inference phase**, presented in Figure 3c, the model's requirement is to handle a large number of query-product pairs. Due to this, the model's scalability and latency become the primary concerns. Hence, we only use the pre-computed node neighborhood (no computation during inference) and limit our compute power to a single GPU. The inference pipeline initiates an instance of the SALAM model and loads the parameters from the latest checkpoint. Subsequently, it processes the query-product pairs and their node neighborhoods (computed during training) received as input to predict the relevance labels and stores them in a database for utilization in the e-commerce engine's relevance computation pipeline. Note that, the inference pipelines for different regions is unique and accordingly utilize the corresponding region-specific model checkpoints.

Further experiments on the evaluation of training and inference pipelines are presented in Sections 4.2 and 4.5, respectively. The training and inference algorithms are detailed in Appendix A.

4 EXPERIMENTAL STUDY

In this section, we describe our experimental setup that analyzes SALAM's performance on the ESCI classification tasks compared to its current alternatives. The experiments intend to answer the following research questions:

RQ1. Does SALAM outperform the current alternatives on the task of search relevance classification?
RQ2. Is the two-phase training with pre-training and fine-tuning better than the alternative training frameworks?
RQ3. What are the considerations in the choice of graph encoders?
RQ4. How does SALAM compare to its alternatives in suitability for an industrial application?

4.1 Datasets and Baselines

The datasets used in our experiments contain $\approx 12M$ manually annotated query-product ESCI pairs collected from the real-world queries of an e-commerce search engine. The pairs are collected from 16 different e-commerce site regions[3] with queries in various languages. No customer information or personally identifiable information were used in collecting this data. We evaluate the performance of our model on the following three different downstream tasks of the e-commerce search engine: (i) Four class (E vs S vs C vs I): applied in complementary item recommendation [12], (ii) Three class (E vs S vs CI): applied in alternate product suggestion [3] and (iii) Two class (E vs SCI): for better precision in product recommendation [4]. Each one of these three variants leads to different class distributions which is adjusted in our model using class weights of the binary cross entropy loss. 20% of each region's dataset is held-out for testing and the other 80% is divided into a training and validation ratio of 4:1 for 5-fold cross validation in our experiments. More details on the various e-commerce regions and the class distributions are given in Appendix B.

For the baselines, we consider the publicly available mBERT [8] and XLM [6] models as well as the multilingual proprietary framework, hereinafter, referred to as *proprietary e-commerce baseline (or) PEBL*. The reason for the baseline choice is the multilingual nature of our experiments. The models are all first trained on a combined training dataset of all the regions for a small number of epochs (4 in our case), and then fine-tuned on the region-specific datasets for a larger number of epochs (20 in our case) to finally get the region-specific checkpoints used in our experimental studies.

4.2 RQ1: Model Performance

In this experiment, we compare SALAM's performance against the baselines on the task of search relevance classification. The candidate models return a single class for each input query-product pair and the performance is evaluated on the metrics of precision,

[3] Australia (AU), Canada (CA), India (IN), Singapore (SG), United Kingdom (UK), United States (US), Japan (JP), Germany (DE), France (FR), Italy (IT), Spain (ES), Netherlands (NL), Turkey (TR), United Arab Emirates (AE), Mexico (MX), and Brazil (BR).

Table 1: Performance comparison of SALAM against the baselines on the task of search relevance classification in different e-commerce regions on four class E-S-C-I classification. The metrics used for evaluation are Precision (P), Recall (R), and F1. The metrics are presented for each class to avoid bias from class distribution and a macro-average over the classes is provided in column M. Due to the proprietary nature of the models, the results shown are relative to the public mBERT model, i.e., a cell value of x implies that the model performs $(1 + x)$ × mBERT. The best results in each segment are highlighted in bold. All the F1-improvements of SALAM over PEBL and mBERT are statistically significant with a p-value threshold of 0.05.

	Region	Australia (AU)					Canada (CA)					India (IN)					Singapore (SG)				
	Class	E	S	C	I	M	E	S	C	I	M	E	S	C	I	M	E	S	C	I	M
P	mBERT	.000	.000	.000	.000	.000	.000	.000	.000	**.000**	.000	.000	.000	**.000**	**.000**	.000	.000	.000	.000	**.000**	.000
	XLM	.107	.085	.119	-.02	.072	.074	.092	.147	-.06	.060	.065	-.06	-.02	-.19	-.05	.111	.454	.086	-.06	.097
	PEBL	.213	**.170**	.238	-.04	.143	.147	**.183**	**.293**	-.11	**.119**	**.130**	-.13	-.04	-.37	-.10	.222	**.908**	**.172**	-.11	**.193**
	SALAM	**.261**	.097	**.256**	**.009**	**.155**	**.163**	.153	.183	-.02	.116	.116	**.021**	-.00	-.04	**.025**	**.354**	.555	.029	-.08	.150
R	mBERT	**.000**	.000	.000	.000	.000	**.000**	.000	.000	.000	.000	**.000**	.000	.000	.000	.000	**.000**	.000	.000	.000	.000
	XLM	-.01	.788	.817	.774	.332	-.01	.601	1.654	1.416	.382	-.04	.277	.948	.723	.235	-.01	.520	.490	1.417	.289
	PEBL	-.02	1.58	1.63	1.55	.664	-.02	1.20	3.31	**2.83**	.764	-.07	**.553**	1.90	1.45	.470	-.02	1.04	.979	2.83	.578
	SALAM	-.04	**2.00**	**2.05**	**1.77**	**.801**	-.03	**1.36**	**4.47**	2.78	**.872**	-.02	.503	**2.12**	**1.68**	**.533**	-.08	**1.77**	**1.51**	**3.32**	**.791**
F1	mBERT	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
	XLM	.051	.476	.510	.400	.282	.033	.367	1.047	.702	.328	.015	.101	.510	.246	.149	.056	.492	.320	.765	.296
	PEBL	.102	.952	1.02	.800	.563	.065	.733	2.09	1.40	.656	.029	.202	1.02	.491	.298	.111	.984	.639	1.53	.591
	SALAM	**.111**	**1.05**	**1.20**	**.923**	**.644**	**.067**	**.775**	**2.40**	**1.50**	**.716**	**.049**	**.276**	**1.14**	**.912**	**.408**	**.138**	**1.10**	**.752**	**1.74**	**.680**
		United Kingdom (UK)					United States (US)					Japan (JP)					Germany (DE)				
		E	S	C	I	M	E	S	C	I	M	E	S	C	I	M	E	S	C	I	M
P	mBERT	.000	.000	**.000**	**.000**	.000	.000	**.000**	**.000**	**.000**	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
	XLM	.043	-.02	-.01	-.28	-.07	.055	-.07	-.00	-.28	-.07	.075	.394	.121	-.06	.105	.089	.065	.034	-.02	.043
	PEBL	.085	-.05	-.01	-.57	-.13	.110	-.14	-.00	-.57	-.14	.149	.788	.242	-.12	.210	.177	.129	.067	-.04	.086
	SALAM	**.086**	**.138**	-.01	-.10	**.028**	**.134**	-.03	-.04	-.08	**.005**	**.193**	**1.15**	**.633**	**.517**	**.517**	**.190**	**.103**	**.303**	**.119**	**.177**
R	mBERT	**.000**	.000	.000	.000	.000	**.000**	.000	.000	.000	.000	**.000**	.000	.000	.000	.000	.000	.000	.000	.000	.000
	XLM	-.04	.205	2.497	3.466	.300	-.04	.354	1.114	.906	.220	-.01	.537	.941	2.512	.335	-.02	.721	1.196	1.090	.358
	PEBL	-.07	.409	4.99	**6.93**	.599	-.08	.708	2.23	**1.81**	.440	-.02	1.07	1.88	5.02	.669	-.05	1.44	**2.39**	2.18	.715
	SALAM	-.03	**.818**	**8.49**	5.52	**.780**	-.05	**1.20**	**5.50**	1.46	**.701**	**.035**	**1.71**	**3.59**	**6.28**	**1.07**	-.04	**1.59**	2.04	**2.67**	**.749**
F1	mBERT	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
	XLM	.003	.098	1.554	.962	.210	.007	.150	.725	.130	.125	.031	.469	.598	1.070	.295	.034	.405	.611	.598	.288
	PEBL	.005	.196	3.11	1.92	.419	.013	.300	1.45	.260	.249	.062	.937	1.20	2.14	.590	.068	.809	1.22	1.20	.575
	SALAM	**.028**	**.487**	**4.34**	**3.02**	**.685**	**.040**	**.570**	**2.58**	**.784**	**.514**	**.115**	**1.43**	**2.22**	**3.57**	**.996**	**.077**	**.843**	**1.30**	**1.55**	**.650**
		France (FR)					Italy (IT)					Spain (ES)					Netherlands (NL)				
		E	S	C	I	M	E	S	C	I	M	E	S	C	I	M	E	S	C	I	M
P	mBERT	.000	**.000**	**.000**	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	**.000**	.000
	XLM	.036	-.03	.014	-.16	-.03	.088	.007	-.08	-.11	-.02	.069	.116	.103	.000	.069	.100	.127	.100	-.08	.051
	PEBL	.072	-.07	.03	-.32	-.06	.176	.014	-.17	-.21	-.04	.137	**.231**	**.205**	.000	**.138**	.200	**.254**	.200	-.16	.101
	SALAM	**.114**	-.03	-.08	**.076**	**.023**	**.218**	**.080**	**.036**	**.149**	**.123**	**.138**	.230	.149	**.033**	.134	**.285**	.238	**.246**	-.02	**.175**
R	mBERT	**.000**	.000	.000	.000	.000	**.000**	.000	.000	.000	.000	**.000**	.000	.000	.000	.000	**.000**	.000	.000	.000	.000
	XLM	-.02	.253	1.101	.739	.197	-.04	.703	1.291	1.113	.314	-.01	.466	.959	.856	.319	-.03	.285	.702	1.235	.256
	PEBL	-.04	.505	2.20	1.48	.393	-.08	1.41	2.58	2.23	.627	-.02	**.931**	1.92	1.71	.637	-.07	.569	1.40	2.47	.512
	SALAM	-.04	**1.11**	**3.92**	**1.92**	**.678**	-.06	**1.93**	**3.77**	**2.65**	**.875**	-.03	.913	**2.01**	**1.89**	**.663**	-.07	**1.24**	**1.89**	**3.01**	**.753**
F1	mBERT	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
	XLM	.010	.138	.748	.288	.163	.026	.384	.641	.491	.252	.030	.296	.534	.460	.245	.037	.213	.430	.609	.236
	PEBL	.020	.275	1.50	.576	.326	.051	.767	1.28	.981	.504	**.059**	**.591**	**1.07**	.919	.489	.073	.425	.860	1.22	.471
	SALAM	**.040**	**.573**	**2.11**	**1.13**	**.558**	**.083**	**1.02**	**1.94**	**1.53**	**.743**	.056	.583	1.05	**1.01**	**.499**	**.110**	**.688**	**1.09**	**1.58**	**.639**
		Turkey (TR)					United Arab Emirates (AE)					Mexico (MX)					Brazil (BR)				
		E	S	C	I	M	E	S	C	I	M	E	S	C	I	M	E	S	C	I	M
P	mBERT	.000	.000	.000	.000	.000	.000	.000	.000	**.000**	.000	.000	.000	.000	.000	.000	.000	.000	.000	**.000**	.000
	XLM	.087	.043	.012	-.03	.030	.106	.115	.098	-.10	.056	.105	.042	.307	-.11	.053	.147	.028	.360	-.05	.090
	PEBL	.173	**.085**	.023	-.05	.059	**.212**	**.230**	**.196**	-.20	.112	.210	**.083**	.614	-.22	.106	**.294**	.055	.719	-.11	.179
	SALAM	**.220**	.018	**.052**	**.074**	**.091**	.209	.208	.103	-.02	**.127**	**.258**	.059	**.686**	**.015**	**.195**	.291	**.056**	**.746**	-.00	**.214**
R	mBERT	**.000**	.000	.000	.000	.000	**.000**	.000	.000	.000	.000	**.000**	.000	.000	.000	.000	**.000**	.000	.000	.000	.000
	XLM	-.02	.164	.652	.376	.190	-.02	.376	1.074	1.870	.352	-.03	.488	.959	1.345	.314	-.03	.710	1.155	.618	.330
	PEBL	-.04	.327	1.30	.752	.380	-.04	.752	2.15	**3.74**	.703	-.06	.976	1.92	2.69	.628	-.05	1.42	2.31	**1.24**	.659
	SALAM	-.03	**.468**	**1.57**	**.864**	**.474**	-.03	**.836**	**2.88**	2.94	**.739**	-.06	**1.61**	**2.61**	**2.88**	**.832**	-.04	**1.51**	**2.42**	1.13	**.678**
F1	mBERT	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
	XLM	.036	.106	.342	.185	.142	.045	.254	.660	.773	.288	.040	.293	.666	.598	.271	.062	.368	.795	.290	.276
	PEBL	.072	.212	.683	.369	.283	**.090**	.508	1.32	1.55	.575	.080	.585	1.33	1.20	.542	.123	.736	1.59	.579	.551
	SALAM	**.095**	**.236**	**.804**	**.499**	**.346**	**.090**	**.532**	**1.50**	**1.58**	**.614**	**.103**	**.813**	**1.67**	**1.55**	**.707**	**.129**	**.770**	**1.66**	**.622**	**.580**

recall, and F-score. The results of our experiments on the four class variant of the problem are given in Table 1. Results on other variants follow a similar trend and are provided in Appendix C.

From our results, we observe that SALAM consistently outperforms both the baselines on F1-score across all regions and classes upto 65% (on an average by 15%). While SALAM is able to provide higher precision than the previous baselines for the 'E' class in several regions such as AU, CA, IN, and others, the major boost in overall performance is driven by improvement in performance on the classes of S, C, and I. This demonstrates SALAM's ability to leverage the graph information to learn better discriminative features unaffected by the innate class imbalance of the datasets. Also, we note that the performance improvement for the regions with highest number of data samples, i.e., IN, UK, and US is also the highest at 11%, 23%, and 24%, respectively. This implies that SALAM is able to learn better discriminative features with higher number

of training samples. Furthermore, we note that the average performance improvement on majority English language e-commerce regions is 13.5% and non-English language e-commerce regions is 7.9%. However, comparing the regions of English CA and non-English TR with similar number of data samples, we observe that the performance improvement is comparable at $\approx 6\%$. Hence, we conclude that, while the stand-alone performance is high, the reason for lower performance boost in non-English language regions is caused by a lower number of data samples and not due to the model architecture. Another point to note is that the SALAM's performance is $30\% - 80\%$ higher than XLM. Given that XLM is the language model used in SALAM, this performance gain demonstrates the main advantage of including graph information in the task of search relevance classification.

4.3 RQ2: Two-phase Learning Framework

To confirm the effectiveness of the two-phase learning framework with region-agnostic pre-training and region-specific fine-tuning, we compare it against a single phase training and validation approach. To get the single phase model (SALAM-S), we train the model on the combined training datasets of all the models but for a larger number of epochs (24 in this experiment)[4]. The comparison metrics are precision, recall, and F1. The results of this experiment are presented in Table 2.

Table 2: Loss in performance by using a single phase training framework (SALAM-S) in comparison to the two phase training model. The metrics are relative to the performance of SALAM, i.e., a cell value of x implies that SALAM-S performs $(1 + x) \times$ SALAM.

Region	P	R	F1	Region	P	R	F1
AU	-.154	-.110	-.150	**FR**	-.054	-.265	-.207
CA	-.080	-.197	-.153	**IT**	-.129	-.153	-.148
IN	-.107	-.089	-.117	**ES**	-.085	-.117	-.106
SG	-.097	-.062	-.080	**NL**	-.070	-.134	-.107
UK	-.030	-.175	-.124	**TR**	-.040	-.097	-.071
US	-.055	-.155	-.121	**AE**	-.038	-.072	-.061
JP	-.159	-.182	-.174	**MX**	-.069	-.114	-.102
DE	-.103	-.212	-.171	**BR**	-.076	-.070	-.073

From our results, we observe that two-phase learning improves the performance on test set by 3%-27% when compared to a single-phase framework trained on the same number of epochs. This exhibits the unique nature of each region's e-commerce queries and promotes the case for region-specific fine-tuning even in the case of multilingual models.

4.4 RQ3: Graph Encoders

For this experiment, we select the popular graph representation frameworks of DeepWalk, Chebyshev, Simple Graph Neural Network (GNN), Graph Recurrent Network (GRN), Graph Convolution Networks (GCN), and Graph Attention Network (GAT) as our candidates. We replace the neighborhood aggregation module with the candidate graph networks in our comparative analysis. To select the graph encoder, we primarily consider the performance metrics of precision, recall, and F1, but in a computationally restrained (to 4GB of VRAM) setting to simulate industry setup. The results of our study are presented in Table 3.

Table 3: Comparison between variants of SALAM with different graph encoders. The results are shown relative to the performance of the DeepWalk variant, i.e., a cell value of x implies that the variant performs $(1 + x) \times$ DeepWalk.

Model Variant	P	R	F1
SALAM-**DeepWalk** [25]	.000	.000	.000
SALAM-**GNN** [34]	.126	.498	.327
SALAM-**GRN** [39]	.167	.557	.377
SALAM-**GAT** [31]	.208	.616	.427
SALAM-**Chebyshev** [7]	.213	.627	.435
SALAM-**GCN** [15]	**.235**	**.678**	**.472**

In our analysis, we observe that given the industry-level memory constraints, the GCN network performs the best in the classification task and hence, we select it as the graph encoder in SALAM.

4.5 RQ4: Practical Environment

In this section, we analyze the memory requirements and processing speed of the model's training and inference pipeline and compare it against the baselines for its suitability to an industrial application[5].

Table 4: Memory and processing requirements of different models in the training and inference pipeline. The columns present the number of parameters (Param), pre-training time (PTT), fine-tuning time (FTT), inference time (IT), VRAM requirement (Mem), and disk space requirement (Disk). PTT and FTT are provided in seconds per epoch and IT is given in milliseconds per sample. Mem and Disk requirements are given in Gigabytes.

Model	Param	PTT	FTT	IT	Mem	Disk
mBERT	110M	128K	215	10.67	1.01	16.2
XLM	270M	133K	229	10.87	2.49	39.8
PEBL	150M	130K	217	10.83	1.38	22.1
SALAM	279M	135K	231	11.05	2.57	169.1

From Table 4, we note that our model and the baselines have similar memory requirements to the XLM model and conforms to the constraints of a general industrial application environment. Additionally, we note that the increase in training and inference times of SALAM when compared to PEBL is within $\approx 3\%$ and $\approx 2\%$, respectively. Given the monthly build timelines, the difference is negligible even for a larger scale of data. However, a significant challenge is the graph pre-computation module. Given, the large scale of product catalogues and the frequency of queries, it is computationally challenging to retrieve the local neighborhood during inference. Hence, we utilize the training set and purchase information to pre-compute the local neighborhoods and store it as a hash-table for $O(1)$ retrieval during inference. The inductive nature of SALAM ensures language model-level results are available for the non-covered infrequent queries. However, when compared to PEBL, the graph adds to a huge disk requirement (128GB) and additional maintenance in practice.

[4]Note that the two phase approach uses 4 epochs for pre-training and 20 for finetuning, so to maintain a fair comparison we train the single phase with 24 epochs.

[5]generally 4GB of VRAM, 16 GB RAM, and 1TB of disk space.

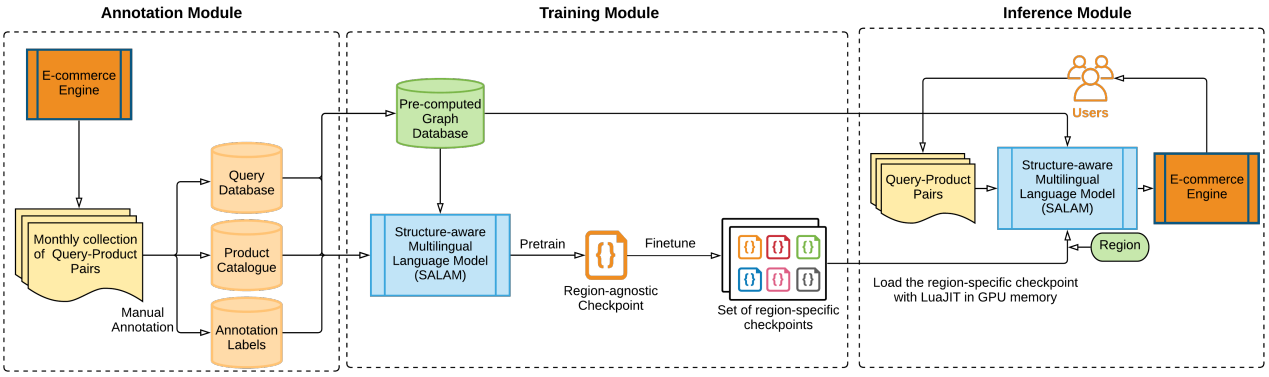


Figure 4: SALAM Workflow. The existing annotation module can seamlessly be integrated with the training and inference module of our model. The additional disk requirement for the pre-computed graph database is trivial and within limits of general industrial settings.

5 DISCUSSION

In this section, we describe the strategy to deploy SALAM in an industrial application and discuss its impact to e-commerce search.

5.1 Deployment Strategy

As illustrated in Figure 4, SALAM can be trained offline using manually annotated datasets and region-specific checkpoints can be stored for inference. One can load these checkpoints to unique SALAM instances for the different e-commerce regions and deploy the model in a industrial setting using the LuaJIT[6] platform. LuaJIT loads the model in-memory for real-time inference, removing the additional overhead of weight transfer from disk to GPU memory. In accordance with the inference pipeline, query-product pairs can be processed in batches on multiple processing units and the labels saved for further utilization in downstream modules. For any workflow that already uses query-product pair as an input for classification, integration of SALAM with such workflows is trivial with minor changes. The graph preprocessing operation is also cached and looked up in real-time. Specifically, given the long tailed nature of product search queries, one can precompute query and product neighborhoods for the samples that correspond to the majority of traffic. Efficient online inference methods can also be used to compute the representations in real-time for the tail. Given the performance gains from the graph information and negligible difference in inference time, deploying SALAM makes it a viable investment in practice.

5.2 Contribution to the search engine

In this section, we describe the impact of having an ESCI classifier in a product search engine. E-commerce product search ranking depends on lexical, behavioral, and semantic matching. Behavioral data can sometimes be noisy, and this leads to biases in the data used to train these models (see for e.g., [20]). Having a classifier perform an ESCI-based classification at the final stage, or use these as inputs to other downstream models can help reduce these issues. Results from these models can also be used for customer messaging as to why a certain item was shown in response to a query; for example, why a charger was shown for a phone query. Improving the performance of ESCI classifiers can thus help improve both product search quality and the trust customers place in these systems, helping practitioners better serve their customers in turn.

[6]https://luajit.org/

6 CONCLUSION

In this paper, we presented Structure-Aware multilingual LAnguage Model (SALAM), a hybrid text and graph based model, that uses a language model in congruence with a GCN network to respectively capture both semantic and relational information for the task of ESCI classification. We have shown the effectiveness of our model in comparison to the currently used alternatives on multiple ESCI classification problem variants. Additionally, we have also demonstrated the effectiveness of a two-phase training setup in handling multiple e-commerce regions, as well as empirically demonstrated our choice of the graph encoder. Moreover, we have also compared SALAM with its alternatives in terms of memory and processing requirements during inference and also detailed its case for deployment in existing industrial pipelines.

REFERENCES

[1] Aman Ahuja, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K. Reddy. 2020. *Language-Agnostic Representation Learning for Product Search on E-Commerce Platforms.* Association for Computing Machinery, New York, NY, USA, 7–15. https://doi.org/10.1145/3336191.3371852

[2] 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.).

[3] Tong Chen, Hongzhi Yin, Guanhua Ye, Zi Huang, Yang Wang, and Meng Wang. 2020. *Try This Instead: Personalized and Interpretable Substitute Recommendation.* Association for Computing Machinery, New York, NY, USA, 891–900. https://doi.org/10.1145/3397271.3401042

[4] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K. Reddy. 2022. ANTHEM: Attentive Hyperbolic Entity Model for Product Search. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining* (Virtual Event, AZ, USA) *(WSDM '22).* Association for Computing Machinery, New York, NY, USA, 161–171. https://doi.org/10.1145/3488560.3498456

[5] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.* Association for Computational Linguistics, Online, 8440–8451. https://doi.org/10.18653/v1/2020.acl-main.747

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

[7] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In

Proceedings of the 30th International Conference on Neural Information Processing Systems (Barcelona, Spain) *(NIPS'16)*. Curran Associates Inc., Red Hook, NY, USA, 3844–3852.

[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] Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. 2020. Scalable Multi-Hop Relational Reasoning for Knowledge-Aware Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 1295–1309. https://doi.org/10.18653/v1/2020.emnlp-main.99

[10] 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

[11] 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.

[12] 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

[13] Marti A Hearst, Anna Divoli, Harendra Guturu, Alex Ksikes, Preslav Nakov, Michael A Wooldridge, and Jerry Ye. 2007. BioText Search Engine: beyond abstract search. *Bioinformatics* 23, 16 (2007), 2196–2197.

[14] Gary King and Langche Zeng. 2001. Logistic regression in rare events data. *Political analysis* 9, 2 (2001), 137–163.

[15] 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. https://openreview.net/forum?id=SJU4ayYgl

[16] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Bkg6RiCqY7

[17] 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.

[18] 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.

[19] 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)*.

[20] Thanh Nguyen, Nikhil Rao, and Karthik Subbian. 2020. Learning Robust Models for e-Commerce Product Search. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 6861–6869. https://doi.org/10.18653/v1/2020.acl-main.614

[21] 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

[22] 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

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019), 8026–8037.

[24] 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

[25] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, New York, USA) *(KDD '14)*. ACM, New York, NY, USA, 701–710. https://doi.org/10.1145/2623330.2623732

[26] 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 arXiv:https://www.science.org/doi/pdf/10.1126/science.290.5500.2323

[27] 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.

[28] 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

[29] 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

[30] 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. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[31] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations* (2018). https://openreview.net/forum?id=rJXMpikCZ

[32] Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A Deep Architecture for Semantic Matching with Multiple Positional Sentence Representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (Phoenix, Arizona) *(AAAI'16)*. AAAI Press, 2835–2841.

[33] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network 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, 1225–1234. https://doi.org/10.1145/2939672.2939753

[34] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48* (New York, NY, USA) *(ICML'16)*. JMLR.org, 40–48.

[35] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph Convolutional Networks for Text Classification. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (Jul. 2019), 7370–7377. https://doi.org/10.1609/aaai.v33i01.33017370

[36] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 535–546. https://doi.org/10.18653/v1/2021.naacl-main.45

[37] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do Transformers Really Perform Bad for Graph Representation? arXiv:2106.05234 [cs.LG]

[38] 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 *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (Eds.). https://openreview.net/forum?id=OeWooOxFwDa

[39] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models.. In *ICML (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 5694–5703. http://dblp.uni-trier.de/db/conf/icml/icml2018.html#YouYRHL18

[40] Jason Zhu, Yanling Cui, Yuming Liu, Hao Sun, Xue Li, Markus Pelger, Tianqi Yang, Liangjie Zhang, Ruofei Zhang, and Huasha Zhao. 2021. TextGNN: Improving Text Encoder via Graph Neural Network in Sponsored Search. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 2848–2857. https://doi.org/10.1145/3442381.3449842

A SALAM ALGORITHMS

Algorithms 1 and 2 are the training and inference process flows of the SALAM model, respectively.

Algorithm 1: SALAM training flow.

Input: Query-product pairs $x = \{(Q, P)\}$, Pre-computed neighborhoods $x_{\mathcal{G}} = \{(Q_{\mathcal{G}}, P_{\mathcal{G}})\}$, Ground truth y;
Output: Predictor P_θ, θ;

Initialize model parameters θ;
for *number of epochs; until convergence* **do**
  Initialize loss $l = 0$;
  for $(Q, P) \in x, (Q_{\mathcal{G}}, P_{\mathcal{G}}) \in x_{\mathcal{G}}$ **do**
    # Process through language model
    $t_{Q,P} = \phi^f_F(XLM(Q, P))$; using Eq. (2)
    # Aggregate node neighborhoods
    $g_{Q,P} = GCN(Q_{\mathcal{G}}) \circ GCN(P_{\mathcal{G}})$; using Eq. (5)
    # Attention Pooling
    $e_{Q,P} = t_{Q,P} \circ g_{Q,P}$; using Eq. (6)
    $q_{Q,P} = \phi^h_{3f}(e_{Q,P})$; using Eq. (7)
    $k_{Q,P} = \phi^h_{3f}(e_{Q,P})$; using Eq. (7)
    $v_{Q,P} = \phi^h_{3f}(e_{Q,P})$; using Eq. (7)
    $a_{Q,P} = \left\{\frac{exp(\alpha_{ij})}{\sum^h_{j=0} exp(\alpha_{ij})} v_i\right\}^h_{i=0}$; using Eq. (8)
    $o_{Q,P} = Softmax\left(\phi^{|y|}_h(a_{Q,P})\right)$; using Eq. (9)
    # Loss Calculation
    $l = l + L(o_{Q,P}, y_k)$; using Eq. (10)
  end
  $\theta \leftarrow \theta - \nabla_\theta l$; # Update parameters
end
return P_θ, θ

Algorithm 2: SALAM inference flow.

Input: $\{(Q, P)\}$, $\{(Q_{\mathcal{G}}, P_{\mathcal{G}})\}$, Region-specific Model Parameters θ_r;
Output: Label y_k;

$t_{Q,P} = \phi^f_{F,\theta_r}(XLM_{\theta_r}(Q, P))$; using Eq. (2)
$g_{Q,P} = GCN_{\theta_r}(Q_{\mathcal{G}}) \circ GCN_{\theta_r}(P_{\mathcal{G}})$; using Eq. (5)
Attention Pooling
$e_{Q,P} = t_{Q,P} \circ g_{Q,P}$; using Eq. (6)
$q_{Q,P} = \phi^h_{3f,\theta_r}(e_{Q,P})$; using Eq. (7)
$k_{Q,P} = \phi^h_{3f,\theta_r}(e_{Q,P})$; using Eq. (7)
$v_{Q,P} = \phi^h_{3f,\theta_r}(e_{Q,P})$; using Eq. (7)
$a_{Q,P} = \left\{\frac{exp(\alpha_{ij})}{\sum^h_{j=0} exp(\alpha_{ij})} v_i\right\}^h_{i=0}$; $\alpha_{ij} = \frac{q_i k_j}{\sqrt{2h}}$; using Eq. (8)
$o_{Q,P} = Softmax\left(\phi^{|y|}_{h,\theta_r}(a_{Q,P})\right)$; using Eq. (9)
$y_k = \arg\max(o_{Q,P})$
return y_k

B DATASET DETAILS

In this section, we study the different details of our dataset which contains a total of $\approx 12M$ annotated samples with E-S-C-I labels. The dataset is subsampled for annotation and does not, in any way, represent the exact query volumes received by e-commerce engines. On an average, 71.8%, 15.1%, 8.1%, and 4.9% of the overall dataset are E, S, C, and I classes. We notice that the *Exact* class is a significantly dominant part of the dataset and there is an apparent class imbalance. Hence, to handle the imbalance we use cross entropy with class weights computed as given in [14].

Figure 5 depicts the distribution of our dataset across the different e-commerce regions. We observe that the regions of US and IN constitute $\approx$ 37% of the total number of samples and consequently, also show higher performance improvements in our experiments (given in Section 4). The rest of regions are similarly distributed of the remaining 63% and hence show comparable performance gains.

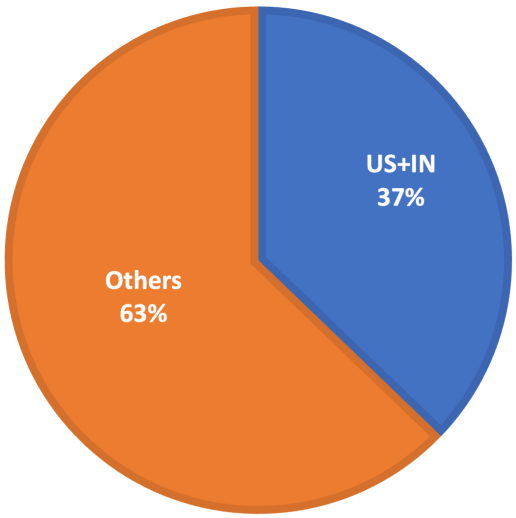


Figure 5: Regional distribution of our dataset. The areas of the chart present the size of different e-commerce regions in the overall composition of our dataset[7].

C FURTHER RESULTS

Table 5 presents results on the three class (E vs S vs CI) and two class (E vs SCI) classification variants of the search relevance problem that are applied to various downstream applications in the e-commerce engine. An example application of the three class problem is alternate product suggestion and that of the two class problem is high precision product search.

[7]The sample distribution of the datasets is given in (%) due to its proprietary nature.

Table 5: Performance comparison of SALAM against the baselines on the task of search relevance classification in different e-commerce regions for three class E-S-CI and two class E-SCI classification. The evaluation metrics are Precision (P), Recall (R), and F1. The metrics are presented for each class to avoid bias from class distribution. Due to the proprietary nature of the models, the results shown are relative to the public mBERT model, i.e., a cell value of x implies that the model performs $(1 + x) \times$ mBERT. The best results in each segment are highlighted in bold. All the F1-improvements of SALAM over PEBL and mBERT are statistically significant with a p-value threshold of 0.05.

	Region	Australia (AU)					Canada (CA)					India (IN)					Singapore (SG)				
	Class	E	S	CI	E	SCI	E	S	CI	E	SCI	E	S	CI	E	SCI	E	S	CI	E	SCI
P	mBERT	.000	**.000**	.000	.000	.000	.000	**.000**	.000	.000	.000	**.000**	.000	.000	.000	.000	.000	**.000**	**.000**	.000	**.000**
	XLM	.020	-.12	-.00	.044	-.03	.022	-.12	-.02	.048	-.03	-.00	-.10	-.04	-.00	-.01	.014	-.27	-.02	.035	-.07
	PEBL	.028	-.05	.020	**.054**	**.028**	**.027**	-.05	.005	**.056**	**.040**	-.00	-.08	-.02	**.001**	.025	.013	-.15	-.01	.043	-.00
	SALAM	**.029**	-.06	**.044**	.035	.025	.018	-.07	**.062**	.046	.029	-.00	**.089**	**.171**	-.00	**.137**	**.100**	-.08	-.03	**.091**	-.02
R	mBERT	**.000**	.000	.000	.000	.000	**.000**	.000	.000	**.000**	.000	.000	.000	**.000**	.000	**.000**	**.000**	.000	.000	**.000**	.000
	XLM	-.03	.102	.014	-.03	.120	-.04	.105	.077	-.03	.198	-.03	-.00	-.04	-.01	-.01	-.05	.064	-.01	-.05	.093
	PEBL	-.01	.118	.042	.005	**.136**	-.02	.109	**.096**	**.000**	**.216**	-.02	-.01	-.07	.012	**.000**	-.02	.077	-.01	-.01	.093
	SALAM	-.02	**.150**	**.055**	**.007**	.089	-.02	**.127**	.041	-.00	.179	**.033**	**.040**	-.00	**.063**	-.02	-.04	**.318**	**.168**	-.03	**.189**
F1	mBERT	**.000**	.000	.000	.000	.000	**.000**	.000	.000	.000	.000	.000	.000	.000	.000	.000	**.000**	.000	.000	.000	.000
	XLM	-.01	-.01	.007	.009	.046	-.01	-.01	.031	.010	.081	-.02	-.06	-.04	-.00	-.01	-.02	-.10	-.01	-.01	.018
	PEBL	-.03	.153	.086	.030	**.084**	-.02	**.119**	**.112**	**.028**	**.128**	-.00	-.06	-.06	.007	.013	-.08	.146	.089	.020	.050
	SALAM	-.03	**.164**	**.105**	**.021**	.058	-.03	.116	**.112**	.022	.105	**.024**	**.044**	**.059**	**.031**	**.058**	-.05	**.325**	**.179**	**.034**	**.089**
		United Kingdom (UK)					United States (US)					Japan (JP)					Germany (DE)				
		E	S	CI	E	SCI	E	S	CI	E	SCI	E	S	CI	E	SCI	E	S	CI	E	SCI
P	mBERT	.000	.000	.000	**.000**	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
	XLM	-.01	-.12	-.06	-.01	-.03	.010	-.10	-.02	.009	-.04	.018	-.06	.038	.035	.004	.020	-.10	.006	.060	-.00
	PEBL	-.01	-.01	-.01	-.01	.048	**.012**	-.04	.021	**.023**	-.01	.021	-.03	.049	.038	.031	**.027**	.005	.033	**.069**	**.071**
	SALAM	**.011**	**.090**	**.491**	-.01	**.233**	.011	**.154**	**.552**	.006	**.194**	**.046**	**.199**	**.613**	**.049**	**.228**	.026	**.007**	**.034**	.052	.067
R	mBERT	.000	.000	**.000**	.000	**.000**	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
	XLM	-.01	-.10	-.08	-.00	-.06	-.03	.083	**.005**	-.03	**.043**	-.02	.099	.064	-.02	.141	-.03	**.093**	.040	-.02	.204
	PEBL	.016	-.13	-.08	.021	-.06	-.01	.100	-.01	-.02	.095	-.01	.105	.074	-.01	.146	-.01	.089	.116	.013	**.221**
	SALAM	**.036**	**.396**	-.12	**.068**	-.09	**.050**	**.285**	-.08	**.068**	.000	**.066**	**.422**	**.202**	**.072**	**.159**	-.01	.072	**.155**	**.015**	.168
F1	mBERT	.000	.000	**.000**	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	**.000**	.000	.000	.000	.000
	XLM	-.01	-.11	-.06	-.01	-.04	-.01	-.01	-.01	-.01	-.00	.000	.022	.050	.006	.068	-.01	-.01	.024	.019	.099
	PEBL	.018	-.01	-.23	.006	-.00	.008	.089	-.16	.002	.038	.008	.122	-.03	.014	.085	-.01	**.098**	.137	.041	**.147**
	SALAM	**.038**	**.327**	-.07	**.027**	**.060**	**.040**	**.292**	**.006**	**.037**	**.095**	**.057**	**.412**	**.275**	**.061**	**.193**	-.02	.090	**.158**	**.034**	.119
		France (FR)					Italy (IT)					Spain (ES)					Netherlands (NL)				
		E	S	CI	E	SCI	E	S	CI	E	SCI	E	S	CI	E	SCI	E	S	CI	E	SCI
P	mBERT	.000	**.000**	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
	XLM	.013	-.17	-.03	.009	-.07	.013	-.08	.057	.024	-.02	.006	-.09	.016	.029	.019	-.01	-.20	-.04	.016	-.06
	PEBL	.011	-.11	-.03	.017	.001	.021	-.01	.089	.033	.031	**.014**	-.04	**.038**	**.040**	.073	.026	-.11	-.01	.058	.008
	SALAM	**.052**	-.02	**.270**	**.049**	**.026**	**.043**	**.055**	**.394**	**.046**	**.099**	-.00	**.021**	.023	.022	**.099**	**.050**	**.085**	**.090**	**.059**	**.080**
R	mBERT	**.000**	.000	.000	**.000**	.000	.000	.000	.000	.000	.000	**.000**	.000	.000	.000	.000	.000	.000	.000	.000	.000
	XLM	-.03	.100	.052	-.03	.068	-.02	.080	.012	-.03	.079	-.02	.041	.009	-.00	.099	-.05	.000	-.01	-.05	.051
	PEBL	-.01	.046	.004	-.01	.102	.004	.094	.032	.004	.093	-.01	**.058**	**.041**	.019	**.129**	-.02	.021	.058	-.01	.120
	SALAM	-.01	**.486**	**.314**	-.01	**.276**	**.033**	**.288**	**.087**	**.038**	**.117**	**.000**	-.01	.039	**.034**	.067	**.019**	**.319**	**.113**	**.042**	**.109**
F1	mBERT	**.000**	.000	.000	.000	.000	.000	.000	.000	.000	.000	**.000**	.000	.000	.000	.000	**.000**	.000	.000	.000	.000
	XLM	-.01	-.02	.010	-.01	.003	-.00	.002	.035	-.00	.027	-.01	-.03	.012	.013	.058	-.03	-.10	-.03	.005	-.04
	PEBL	-.04	.195	.057	.005	.055	-.01	.140	.018	.018	.062	-.01	**.049**	**.061**	**.029**	**.101**	-.03	.048	.074	.046	.023
	SALAM	-.02	**.507**	**.380**	**.019**	**.152**	**.020**	**.280**	**.178**	**.042**	**.108**	-.01	.046	.052	.028	.083	-.00	**.319**	**.153**	**.075**	**.053**
		Turkey (TR)					United Arab Emirates (AE)					Mexico (MX)					Brazil (BR)				
		E	S	CI	E	SCI	E	S	CI	E	SCI	E	S	CI	E	SCI	E	S	CI	E	SCI
P	mBERT	.000	**.000**	.000	.000	.000	.000	**.000**	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
	XLM	.003	-.17	-.01	.032	-.04	.001	-.16	-.04	.014	-.05	.013	-.10	.033	.034	-.03	.002	-.09	-.02	.018	-.01
	PEBL	.028	-.04	.025	**.058**	.037	**.002**	-.12	-.03	**.019**	.011	**.023**	-.05	.047	**.049**	.024	**.012**	-.03	.010	**.034**	**.048**
	SALAM	**.053**	-.04	**.058**	.057	**.068**	.000	-.05	**.064**	.001	**.056**	.017	**.015**	**.192**	.033	**.079**	-.01	**.014**	**.093**	.025	.044
R	mBERT	.000	.000	.000	.000	.000	.000	.000	**.000**	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
	XLM	-.04	.035	-.02	-.04	.065	-.04	.008	-.03	-.03	.038	-.02	.054	**.030**	-.03	.090	-.04	.005	-.00	-.02	.031
	PEBL	**.002**	.025	.053	.017	**.100**	-.02	.014	-.04	.002	.042	.003	.086	.029	.002	**.117**	-.00	.012	**.008**	**.032**	**.050**
	SALAM	.000	**.091**	**.115**	**.040**	.095	**.002**	**.096**	-.05	.031	-.00	**.024**	**.237**	.011	**.038**	.069	**.014**	**.039**	-.02	.031	.036
F1	mBERT	.000	.000	.000	.000	.000	**.000**	.000	**.000**	.000	**.000**	**.000**	.000	.000	.000	**.000**	**.000**	.000	.000	.000	.000
	XLM	-.02	-.07	-.01	-.03	.023	-.02	-.08	-.03	.062	-.05	-.00	-.02	.032	.026	-.07	-.02	-.04	-.01	.030	-.02
	PEBL	-.00	.053	.058	.006	.080	-.02	.019	-.05	.084	-.01	-.01	.174	.008	.050	-.04	-.00	.006	.008	**.063**	**.023**
	SALAM	**.004**	**.085**	**.107**	**.017**	**.094**	-.01	**.099**	-.02	**.090**	-.01	-.00	**.295**	**.065**	**.060**	-.03	-.00	**.041**	**.034**	.058	.014