# HISS: A Novel Hybrid Inference Architecture in Embedding Based Product Sourcing using Knowledge Distillation

Ankith M S
Amazon
Bengaluru, India
ankiths@amazon.com

Sourab Mangrulkar
Amazon
Bengaluru, India
smangrul@amazon.com

Vivek Sembium
Amazon
Bengaluru, India
viveksem@amazon.com

## ABSTRACT

Semantic Sourcing is a well-studied area in web and product search to improve the quality of search results. In the context of Semantic Sourcing in e-commerce search, transformer-based models like BERT (fine-tuned for relevance) can be used to encode the representation of queries into a semantic space where the semantically equivalent entities (i.e., queries for Query Reformulation (QR) or products for direct Semantic Sourcing application) are in the neighbourhood of the given query. Although BERT achieves state-of-the-art performance, this comes at a latency cost to compute the embedding, making it unsuitable for real-time reformulations where a Deep Semantic Search Model (DSSM)- a simple architecture comprised of word embedding layer followed by mean-pool layer, is more suitable. In this work, we demonstrate that (1) applying knowledge distillation to transfer the knowledge from SBERT (BERT fine-tuned for relevance) to DSSM shows improvement in AUC of 2.03% in the query-product relevance task compared to training DSSM directly on the relevance data and (2) **HISS**: **H**ybrid **I**nference architecture in **S**emantic **S**earch: DSSM (from knowledge distillation) if used in conjunction with BERT on alignment loss, shows improvement in AUC of 0.8-1.2% over DSSM (from KD) only model.

## CCS CONCEPTS

• **Information systems** → **Web searching and information discovery**; **Document representation**.

## KEYWORDS

Information Retrieval, Query Reformulation, Semantic Sourcing, Semantic Search, Hybrid Inference

## 1 INTRODUCTION

Sourcing products for large e-commerce systems (such as Amazon, Walmart etc) is a challenging problem due to inconsistency between the user query and product information (title, description etc.). For instance, consider the multilingual marketplace like India, where customers speak a diverse set of languages (Hindi, Tamil, Telugu etc.) while both the marketplace and the catalogue are predominantly in English. In this scenario, sourcing relevant products are challenging due to:

- queries having incorrectly spelled words (e.g. – 'mixer grainder').
- queries having product attributes that are spoken in common parlance but unlikely to be present in product title/description, also referred to as natural language queries (e.g., attribute 'fancy' in query 'fancy kurtis').
- queries having words from a vernacular language (e.g – 'jeera' – substitute for 'cumin seeds' in English).

The above challenges are further intensified in the real-time scenario, where customers expect relevant products to be retrieved in just a fraction of a second.

This problem can be naturally solved using two approaches (1) Query Reformulation (QR) - In this approach, customer query not so rich in relevant products can be reformulated to a semantically equivalent query that contains a rich set of relevant products (2) Semantic Sourcing (SS) - given a customer query, directly retrieve K nearest neighbour products form the shared embedding space of query and products. Although QR is an efficient approach to retrieving semantically equivalent products without changing the indexing system, this approach does not work well on tail queries where the queries are unique, rare, and comprise a considerable amount of traffic. On these queries, it will be more effective to source products directly using the SS approach compared to QR. In this paper, we limit our discussions to SS, although our methods are equally effective and applicable to QR as well. Note that hosting an online SS system requires building (1) **semantic index** - large scale indexing system to index products based on the product representation (2) **query processor** - where a given customer query is converted to a query representation in real-time and (3) **KNN search** - the query is matched to nearest neighbour products in real-time. In this paper, we focus on learning better representation models for both queries and products so that they can be used for SS and omit details of engineering systems.

Our approach to semantic match is to represent queries and products independently into an n-dimensional semantic space where the semantically equivalent queries (or) products can be found in the neighbourhood of the semantic representation of the query under consideration. Transformer models such as BERT[8] have recently become ubiquitous in NLP applications and has been successfully applied to the Search Sourcing problem [26, 6, 2]. The success of transformers has been primarily attributed to self-supervised learning and self-attention mechanism. The self-supervised learning approach improves the model due to the training on abundantly available unsupervised datasets while the self-attention mechanism increases the complexity of the models. We fine-tune BERT (called SBERT)[26] to independently represent queries and products into a rich n-dimensional semantic space. Our SBERT model can be used


*SIGIR eCom'22, July 15, 2022, Madrid, Spain*

to implement offline sourcing strategies like caching of products to be sourced for popular queries either by Query Reformulation or KNN product search and retrieving them in real-time from the cache during online serving. However, for real-time SS, the query processor has a low latency requirement making the transformer based models infeasible to use.

Nigam et al. introduced DSSM [25], a simple architecture comprised of word embedding layer followed by mean-pool layer, which is more suitable for real-time scenarios with low latency requirements. Although this model can be leveraged in place of transformer models to represent queries or products, it suffers from performance due to the lack of self-supervised learning and complexity in representation. In order to bridge the gap, we leverage Knowledge Distillation (KD) [12, 38] techniques where we use DSSM as a student model to learn the rich semantic representation from the SBERT model by (1) imitating the soft relevance probabilities from the teacher model instead of hard ground truth labels and (2) aligning the representation from DSSM and BERT models for interoperability of representations. This process enables the transfer of rich semantic knowledge from the high capacity teacher model to the low latency student model that cannot be trained using self-supervision.

Once the student model is trained using the KD technique, the high capacity teacher model is typically discarded in the inference pipeline. However, in this work, we empirically demonstrate that a student model amalgamated with the teacher model in the real-time sourcing applications outperforms a student-only inference network.

**Our contributions** are as follows. 1) We propose a methodology to distil the query-product relevance knowledge encoded in BERT to low latency simple architecture like DSSM, which can be further used to represent entities (query and product) in a shared embedding space for sourcing applications. 2) We propose an alignment loss that enforces the student and teacher network to map the queries (and product) representation to a model-agnostic embedding space. 3) We also propose a **HISS**: **H**ybrid **I**nference architecture in **S**emantic **S**earch- a blend of highly accurate teacher model-SBERT and low latency student model-DSSM, achieving the best of both worlds for real-time retrieval systems. Our experiments show that the heterogeneous model outperforms the DSSM(from KD) only model by 0.8-1.2%.

## 2 RELATED WORK

**Embedding based retrieval or Dense retrieval** Word2Vec[23] opened up a new era in the NLP by representing words in a dense vector such that two semantically similar words will have similar representation. Post that, many works have been carried out in semantic sourcing to represent queries and documents into a dense vector and leverage them for document retrieval. Some related works in this space are DSSM [25, 14], CDSSM [30],Multi-Task DNN [19]. Recent advances in pre-trained language models like BERT[8], RoBERTa [20] achieved state-of-the-art in various NLP tasks. Sentence-BERT [26] became popular in Information Retrieval domain.

**Query Reformulation** (QR) is a well-studied research area in Information Retrieval(IR) to enable semantic sourcing. The works in this area can be classified into the following categories (but are not limited to these) i) Term expansion [41, 5, 37] ii) Term dropping and substitution [4, 17] iii) Machine Translation [27, 40] iv) Reinforcement Learning [24] and v) Representation learning [32, 9, 11]. Representation learning based QR is one of the embedding based product retrieval methods.

**Efficient Transformers** Although transformer models like BERT accomplished state-of-the-art in various NLP tasks, they cannot be deployed in real-time sourcing applications because of their high inference cost. Hence, in the last two years, building an efficient transformer model has been an active research area. These efficient transformer models fall under the following 6 categories (i) early exit[42, 36], (ii) knowledge diffusion[10, 7], (iii) efficient self-attention[33], (iv) architectural innovations[26, 16], (v) knowledge distillation[12, 28], and (vi) model compression[3].

**Knowledge Distillation** In the past few years, the neural IR community has proposed various network architectures for effective document retrieval and ranking using knowledge distillation(KD)[12]. Some KD methods such as TinyBert[15], DisitlBERT[28], MiniLM[34] are also proposed to reduce the computation cost especially in BERT. Also, the recent works in KD based dense retrieval such as topic aware sampling(TAS)[13] and TwinBERT [21] have shown a drastic drop in latency for the retrieval task when compared to the baseline BERT.

Even with these advances, efficient transformer and knowledge distilled models are unsuitable for large-scale real-time applications because they require GPU(s) for millisecond inference. In order to overcome the utilization of high compute resource, we propose a novel KD framework comprising alignment loss to distil the knowledge from BERT to low latency architecture (inspired by feature matching loss in [39]). We also propose **HISS** inference network where we retrieve semantically similar documents in less than 5ms (on CPU) but with a slight drop in performance(AUC) metric.

## 3 OUR APPROACH

In this section, we describe our proposed knowledge distillation approach. Here, the objective is to learn the student model's parameters such that the student model's performance is close to that of the teacher but with a significant reduction in inference time. However, to apply knowledge distillation, first, we must learn our teacher model's parameters for the semantic sourcing task. While our objective is to source semantically relevant products for a given query from the shared representation space, our proposed model is robust enough to also be used for the Query Reformulation task. The motivation for us to build a multi-application model is to reduce maintenance/production related costs.

In QR, our objective is to map a given query to several semantically equivalent queries, but we do not have access to supervised query-query pertinent pairs. Instead, we have a copious amount of user behavioural data like purchases, clicks, glance views and also have access to human audited query-product relevance data. In section 3.2 we introduce various loss functions suitable for the above data to learn the query-product relevance in the teacher model,

which is leveraged to generate query-query similarity. Further, in section 3.2.3 we exploit the semantics in product categorization, such as browse taxonomy, to control relevance in the teacher model.

In section 3.4 we introduce our KD-DSSM method- to map queries and products into a model-agnostic embedding space where semantically equivalent entity pairs are closer and irrelevant entity pairs are compelled to be apart. Entity pairs here include query-query, query-product and product-product. Furthermore, we introduce alignment loss to enable model interoperability, this loss enforces the embedding generated by student and teacher models for the same entity to be close to each other in the semantic space .

Finally, in section 3.5, we explain the **HISS**-heterogeneous inference architecture, that leverages model interoperability and combines the best of both (the Student and Teacher) models to build a real-time inference pipeline.

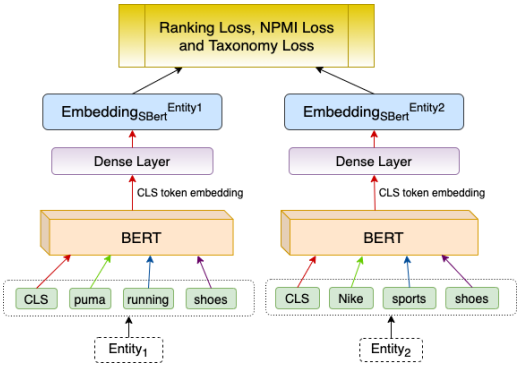


**Figure 1: Teacher Model Architecture-SBERT**

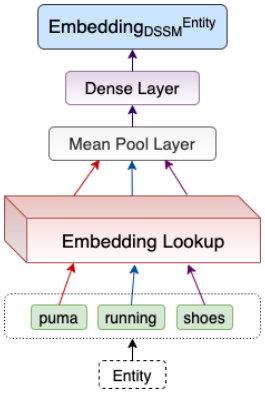


**Figure 2: Student Model Architecture-DSSM**

**Teacher Model:** Figure 1 depicts the Siamese BERT(SBERT)- teacher model architecture. SBERT first computes fixed size contextual representation for an entity using the BERT model's 'cls' token output, followed by a dense layer with 'tanh' activation to get entity embedding $emb^i$. We use the same BERT model for representing both query and product to enable the transfer of language semantics between them. Finally, the similarity $\hat{y_i}$ between entities is determined by the cosine distance between the embeddings. Note that, although we use BERT, any transformer model can act as an alternative to the BERT model.

**Student Model:** Figure 2 depicts the Deep Semantic Search Model (DSSM) - student model architecture. The high-level architecture of DSSM is similar to that of SBERT. The only difference is that, we use embedding lookup and mean-pool layer instead of BERT encoder to generate the intermediate representation. We use the same DSSM model to generate embeddings for the query and product like in SBERT.

## 3.1 Problem Definition

We have SBERT-Teacher and DSSM-student represented by S and KD, respectively. The SBERT model parameters are characterized by a function $S(x;\theta)$ where $x$ is entity (query or product) and $\theta$ represents model parameters. Our objectives in KD-DSSM are to i) learn new parameters $\theta'$ corresponding to the DSSM model $KD(x;\theta')$, such that the performance of KD-DSSM on relevance classification task is close to that of SBERT. ii) Enable model interoperability to build hybrid inference architecture (HISS).

## 3.2 Teacher Training Objective

Let's assume our training samples are represented by $(e_i^1, e_i^2, y_i)$, where $e_i^1$ is a query entity and $e_i^2$ can be query or product entity depending on the data source, and $y_i$ is the ground truth label. Let $cos(v^i, v^j)$ be the function that returns the cosine similarity between two vectors $v^i$ and $v^j$. We devise the following loss functions by exploiting the cosine similarity $\hat{y_i} = cos(S(e_i^1), S(e_i^2))$ to encode the semantics in the model.

*3.2.1* ***Ranking Loss.*** In this step, we leverage the data generated by human annotators who classify the query-product pairs into the following three classes based on relevance: i) Strictly Relevant ii) Standard Relevant, and iii) Irrelevant. Let's call this dataset as $D_{audit}$. The fundamental idea behind ranking loss is, an embedding space should capture relevance gradation. For instance, the query "*Puma shoes*" in the semantic space should be closer to a strictly relevant products such as "*Puma running shoes*" compared to a standard relevant products such as "*Nike Shoes*". This gradation of relevance ensures strictly relevant products are prioritized over the standard relevant products when available. Since query and product share the SBERT model, it transfers the relevance gradation to query-query similarity. The ranking loss is defined in equation 1, where $\theta_{smin}$ and $\theta_{smax}$ are hyperparamters.

$$LL_{RL} = \sum_{(e_i^1, e_i^2, y_i) \in D_{audit}} \Big( \mathbb{1}_{y_i = strict} \ (\hat{y_i} - 1)^2 + \mathbb{1}_{y_i = standard} (min((0, \hat{y_i} - \theta_{smin}))^2 + (max(0, \hat{y_i} - \theta_{smax}))^2) + \mathbb{1}_{y_i = irrelevant} (\hat{y_i})^2 \Big) \quad (1)$$

*3.2.2* ***NPMI Based Loss.*** Generating human audited relevance data is time consuming and expensive process. Practically, it is not feasible to generate the audit data to cover the entire semantic space of e-commerce. However, we have copious amounts of customer behaviour data (search query followed by purchase) which implicitly contains the relevance signal. Let us call this dataset as

$D_{purchase} = \{(q1, p1, c1), (q2, p2, c2)..\}$ where $c_i$ represents the total number of purchases made by customers after firing a query $q_i$ and bought product $p_i$. Note that, although the customer data is abundant, it is noisy and has to be applied in conjunction with the relevance audit data to build a strong relevance model.

Inspired by Laus[18] to use Normalized Point-wise Mutual Information(NPMI) to measure topic co-occurrence, we apply NPMI metric to construct query-query pertinent pairs by leveraging $D_{purchase}$ data. We measure how likely we are to see the two queries co-occur, given their individual probabilities, and compare to the case when the two queries are entirely independent. A probability distribution across queries can be calculated by normalizing the purchase count in $D_{purchase}$. The joint distribution of any given two queries is measured by the strength of common products between them. Leveraging this definition, we construct the semantically similar query-query data $D_{QQ+}$ using the query-product customer data having NPMI (equation 2) scores greater than $\tau_{npmi}$. In Table 1, we show a few examples obtained through this process.

**Table 1: Few QQ positive pairs generated by applying the NPMI (described in section 4.1) on user behaviour data. This data helps us to capture semantics between entities even when there are no common terms between them.**

| Query 1 | Query 2 |
|---|---|
| payal for women | women anklets |
| puma shoes | puma sneaker |
| necklace | wedding jewellery |
| omedone extractor | white heads remover |
| range extender | wifi repeater booster |
| stylus | touch pen |

$$NPMI(q_i, q_j) = \frac{\log \frac{P(q_i, q_j)}{P(q_i), P(q_j)}}{-\log P(q_i, q_j)} \tag{2}$$

$$P(q_i, q_j) = \sum_{k=0}^{Z} \frac{PC(q_i, p_k)}{\sum_{y=0}^{Z} PC(q_i, p_y)} \cdot \frac{PC(q_j, p_k)}{\sum_{y=0}^{Z} PC(q_j, p_y)} \tag{3}$$

$PC$ is the function which returns the purchase count from $D_{purchase}$ for a given query $q_i$ and product $p_j$ pair. $Z$ represents the total number of distinct products in $D_{purchase}$. Leveraging the $D_{QQ+}$ data, we define the following loss function to learn $Query-Query$ semantics, where $\theta_{QQ+}$ is a hyperparameter and $\hat{y_i} = cos(S(e_i^1), S(e_i^2))$.

$$L_{QQ+} = \sum_{(e_i^1, e_i^2) \in D_{QQ+}} (min(0, \hat{y_i} - \theta_{QQ+}))^2 \tag{4}$$

Typically, we set $\theta_{QQ+}$ and $\theta_{smin}$ to same value. Note that in the loss function 4, the cosine score $\hat{y_i}$ is not bound with an upper limit, unlike the loss defined for the standard relevant pair in equation 1. The fundamental intuition behind this loss function is that the pertinent query-pairs in $D_{QQ+}$ does not express the gradation in relevance(Strict vs Standard relevant).

*3.2.3* ***Taxonomy Based Loss.*** E-commerce websites categorize the billions of products available within their service into predefined multi-level product taxonomy or browse nodes. This taxonomy encodes relevance among products and can be exploited to infer various relationships among them. In addition to that, many e-commerce companies have built query classification model [29, 31]. Let us call this model Q2BN, which assigns distribution scores for a query over the taxonomy tree expressed above, and let the dataset generated using Q2BN be $D_{cat} = \{(q_1, bn_1^1, bn_1^2..bn_1^j), (q_2, bn_2^1, bn_2^2..bn_2^l), ..\}$, where $q_i$ is a query and $bn_i^z$ is a browse node corresponding to $q_i$. For instance, query "puma women shoes" will be classified as $bn_1$="shoes->women shoes->running shoes", $bn_2$= "shoes->women shoes->boots" and $bn_3$="shoes->women shoes->casual shoes".

Here, two queries expressing two different intents will have entirely different scores over the taxonomy tree. We found this information particularly useful for recovering user intent with search queries containing ambiguous tokens, for example, *Car bumpers vs Gym bumpers*. We leverage this intuition to generate a set $D_{QQ-}$ (explained in section 3.3 ) of query pairs where the queries involved in a pair express different intents as per the browse association in $D_{cat}$ and define the following taxonomy loss to encode this intent disparity as follows.

$$L_{QQ-} = \sum_{(e_i^1, e_i^2) \in D_{QQ-}} \hat{y_i}^2 \tag{5}$$

Where $D_{QQ-}$ is the query-query negative dataset and $\hat{y_i} = cos(S(e_i^1), S(e_i^2))$.

## 3.3 Training

To learn the semantics in teacher model, we first initialize our BERT model with pretrained weights. This serves as a good initialization point for our model. Further, for the first epoch we leverage the $D_{audit}$ and $D_{QQ+}$ (generated from user behaviour data) to learn our model parameters by optimizing for loss terms in equation 1 and 4.

$$L_1 = \alpha_1 L_{RL} + \alpha_2 L_{QQ+} \tag{6}$$

where $\alpha_1$ and $\alpha_2$ controls the importance of loss terms $L_{RL}$ and $L_{QQ+}$ respectively.

For the next set of epochs we leverage taxonomy tree which encodes relevance among products as described in section 3.2.3 to generate hard negatives. In each epoch, we find queries that are close in the current embedding space but have no common browse node among them and add them to $D_{QQ-}$ as hard negatives to optimize for the following equation

$$L_2 = \alpha_1 L_{RL} + \alpha_2 L_{QQ+} + \alpha_3 L_{QQ-} \tag{7}$$

Where $\alpha_3$ is a weight scalar, which controls the importance of taxonomy loss.

## 3.4 Knowledge Distillation

Figure 3 shows our proposed Knowledge Distillation-DSSM(KD-DSSM) method. Traditional KD approach forces student model to imitate only prediction output of teacher model. The key idea behind this is that soft probabilities from a teacher model is more

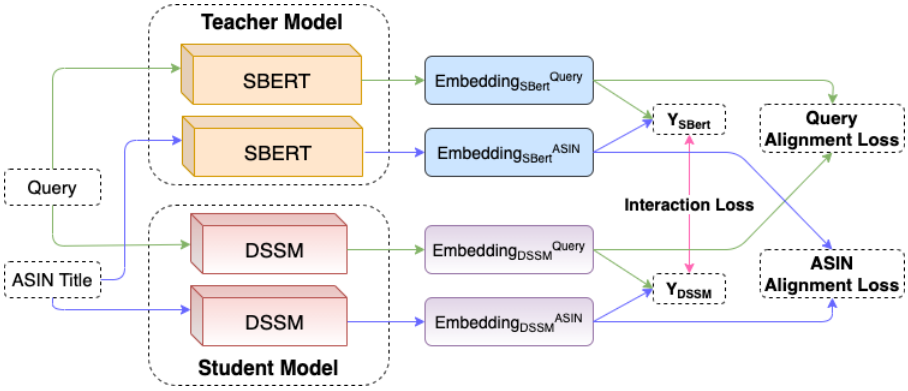


**Figure 3: KD-DSSM: Our proposed knowledge distillation method. i) Alignment loss: Enforces the DSSM model to project embeddings to SBERT embedding space. ii) Interaction loss: Forces DSSM model to mimic SBERT model's prediction**

informative than the hard labels. Unlike traditional KD methods, our student model is trained to (i) Imitate the soft probabilities obtained by the teacher. (ii) Project entities into a teacher representation space by minimizing the cosine loss between final embeddings generated by the student and teacher model for the same entity.

We define the imitation or interaction loss as,

$$L_{imitation}(e_i^1, e_i^2) = -1 * (\hat{y}_i^S \log(\hat{y}_i^D)) \quad (8)$$

Where $\hat{y}_i^S = cos(S(e_i^1), S(e_i^2))$ and $\hat{y}_i^D = cos(D(e_i^1), D(e_i^2))$ are predictions of SBERT $S$ and DSSM $KD$ model respectively.

*3.4.1* ***Training Objective*** . We learn the parameters $\theta'$ of the KD-DSSM model $KD$ by optimizing the following equation 9 given $D_{all} = \{D_{audit} \cup D_{QQ+} \cup D_{QQ-}\}$

$$L_{KD} = \sum_{(e_i^1, e_i^2) \in D_{all}} (L_{imitation}(e_i^1, e_i^2) + L_a(e_i^1) + L_a(e_i^2)) \quad (9)$$

Where $L_a(e_j) = (1 - cos(r_j^S, r_j^{KD}))$ is the alignment loss and $r_j^S = S(e_j)$, $r_j^{KD} = KD(e_j)$ are embeddings generated for an entity $e_j$ by SBERT $S$ and KD-DSSM $KD$ respectively. This ensures that embedding generated by the two models $S$ and $KD$ are close to each other in the model agnostic embedding space for a given entity. The detailed training steps are outlined in Algorithm 1.

## 3.5 HISS: Hybrid Inference Semantic Sourcing Architecture

The key idea of embedding based sourcing is to project queries and products into shared semantic space. When customer fires a query, first convert the incoming query to embedding and fetch the relevant products that are indexed in the space to retrieve them. Broadly, the embedding based retrieval pipeline consists of the following three steps:

- **Semantic Index:** Let $M_1$ be the model that translates the queries into embedding and $D_{products}$ be the set of products. In this step, we leverage HNSW [22] to create an index $I_{HQ}$ of $D_{products}$ using $M_1$, which is further used to retrieve nearest relevant products for a given customer query.
- **Query Processor:** We translate the customer query into embedding $emb_i$ using model $M_2$ in real-time.
- **KNN search:** In this step, we use $emb_i$ as a key to source top-K relevant products from $I_{HQ}$ in real-time.

Typically, the models used during Indexing $M_1$ and Query Processor step $M_2$ step would be the same (either $S$ or $KD$). However, in this work, we show that the heterogeneous inference pipeline ( $M_1 = S(x; \theta)$ and $M_2 = KD(x; \theta')$ ) can be used to leverage the best of $S$ and $KD$. The heterogenous inference pipeline is feasible because

- The creation of semantic index is an offline step with no latency constraint. Therefore, SBERT $S$ which is a high capacity model, despite having a high inference time, can be used to build high quality index.
- During the knowledge distillation process, we optimize for the alignment loss $L_a$ which ensures the embeddings generated by $KD$ and $S$ are close to each other for the same entity in the semantic space. Hence, in query processor step, we can use either $S$ or $KD$ to project query into a shared embedding space.

*3.5.1* ***Other Applications***. The proposed hybrid inference architecture can also be leveraged in other applications as follows.

- **Query Reformulation:** In this approach, when a customer fires a query, we project the query into an embedding space and retrieve top-K relevant queries (which are indexed) using approximate KNN algorithms[22] in real-time.
- **Product to Query sourcing:** This application corresponds to the seller ad service where the seller selects a product/s to participate in the ad auction, and the keyword recommendation engine responds in real-time with the relevant queries the seller can target.

The above applications have three main components i) **Semantic Index**, ii) **Query(for QR) or Product(for keyword recommendation) Processor** and iii) **KNN Search**, similar to query-to-product sourcing pipeline. Hence, we can leverage a highly accurate teacher model for component (i), and low latency student model for component (ii).

**Algorithm 1:** Training KD-DSSM Model

**Require:** $D_{audit}$, $D_{QQ+}$, $D_{cat}$, $N_{epochs}^{teacher-train}$, $N_{epochs}^{student-train}$ and model hyper-parameters

**Initialize:** BERT model parameters in SBERT $S(x;\theta)$ with pre-trained model weights, dense layer in SBERT and DSSM $KD(x;\theta^{'})$ parameters with random weights;

$D_{QQ-}^{all}$ = [];

**Teacher Training**

  update $\theta$ by optimizing loss $L_1$ using the $D_{audit}$ and $D_{QQ+}$ dataset;

  **for** $epoch = 1$ *to* $N_{epochs}^{teacher-train}$ **do**

    Generate $D_{QQ-}$ using embedding from $\theta$ and taxonomy data from $D_{cat}$ ;

    update $\theta$ by optimizing $L_2$ using the $D_{audit}$, $D_{QQ+}$ and $D_{QQ-}$;

    $D_{QQ-}^{all}$ += $D_{QQ-}$;

  **end**

Freeze SBERT - $\theta$ parameters

**Student Training**

  **for** $epoch = 1$ *to* $N_{epochs}^{student-train}$ **do**

    $D_{all} = \{D_{audit} \cup D_{QQ+} \cup D_{QQ-}^{all}\}$;

    get predictions $\hat{y}_i^S$ and $\hat{y}_i^D$ from SBERT $S$ and DSSM $D$ respectively for each pair $(e_i^1, e_i^2) \in D_{all}$;

    get representations $r_i^S$ and $r_i^D$ from $S$ and $D$ respectively for each entity $e_i^1 \in D_{all}$;

    get representations $p_j^S$ and $p_j^D$ from $S$ and $D$ respectively for each entity $e_i^2 \in D_{all}$;

    update $\theta^{'}$ by optimizing $L_{KD}$;

  **end**

# 4 EXPERIMENTS AND RESULTS

## 4.1 Dataset Generation

We collected customer behaviour data during the period between July'20 and Dec'20 from the anonymized session logs of Amazon India marketplace. We are mainly interested in purchase sessions for our use case, where a customer searched for a query 'q' and bought a product 'p'. The number of such independent buying sessions shows the strength of intent similarity between query 'q' and product 'p'. Next, we alleviated noise by omitting all query-product pairs that do not have sufficient strength(<10). Let us call this data $D_{purchase} = \{(q1, p1, c1), (q2, p2, c2)..\}$, where $q_i, p_i$ represents query and product entity, and $c_i$ represents the total number of purchases made by customers after firing a query $q_i$ and bought product $p_i$. We used $D_{purchase}$ to generate $D_{QQ+}$ as described in section 3.2.2

We collected 3M <Query, Product> human audited relevance data $D_{audit}$ from Amazon. Then, we randomly sampled 2 datasets (validation $D_{val}$ and test $D_{test}$) of 30K,20K and 10K query-product pairs belonging to strict,standard and irrelevant labels respectively from relevance judgements, and removed these $120K$ pairs from the training data $D_{audit}$.

We also randomly sampled 500K queries from the distinct queries in $D_{purchase}$ and gathered their browse node associations using the Query Classification Q2BN model. Query to product taxonomy mapping is used as a source to generate $D_{neg}$ to keep irrelevant query-query pairs apart in the embedding space (section 3.2.3).

In the final step of data generation, we randomly sampled 5M unique products $D_{products\_5M}$ from the Amazon catalogue and 100K unique queries $D_{queries\_100K}$ from anonymised customer query logs to report precision@K and latency numbers. Further, we leveraged the Query Classification Q2BN model to generate a query to product taxonomy(or browse node) mapping scores on $D_{queries\_100K}$. We use this dataset to report Query-Query neighbourhood quality.

## 4.2 Experiments

*4.2.1* ***Implementation Details and hyperparameters.*** We implemented all the experiments using Tensorflow [1] and HuggingFace [35]. The backbone for the SBERT model is pre-trained bert-base-uncased [8]. Output embedding dimensions for all experiments were fixed to be 512. Both the models were trained using Adam optimizer with a learning rate of 1e-3 for a maximum of 20 epochs with early-stopping criteria. We set $\tau_{npmi} = 0.45$ to filter out noisy Query-Query similar pairs after applying NPMI on $D_{purchase}$. For teacher training, we set $\theta_{smin}$ and $\theta_{smax}$ to 0.7 and 0.85, respectively and similarly, we set $\theta_{QQ+}$ to 0.7. Along with that, we also set all our loss weight hyperparameters $\alpha$'s to 1. All the experiments were performed on a single GPU on p3.8xlarge EC2 instance on AWS. For training the DSSM (without KD), we used the same set of hyperparameters as in the case of SBERT training. Note that all the hyperparameters were chosen empirically based on the experiments performed.

*4.2.2* ***Results.*** We report the AUC on $D_{test}$ to measure the efficacy of our KD approach on the query-product relevance classification task, and we used DSSM (without KD) as our baseline. Table 2 captures the AUC of teacher and student model on $D_{test}$ dataset. From the table, we can infer that the KD-DSSM model performs better than the DSSM. Hence we can transfer the knowledge encoded in

**Table 2: ROC-AUC and QQ Irrelevance of various models.**

| Model | AUC | QQ Irrelevance |
|---|---|---|
| SBERT-without Taxonomy loss | 0.935 | 24.3% |
| SBERT-with Taxonomy loss | 0.928 | 11.5% |
| DSSM-without Taxonomy loss | 0.892 | 27.08% |
| DSSM-with Taxonomy loss | 0.886 | 14.2% |
| KD-DSSM | 0.9038 | 12.45% |

BERT to smaller networks like DSSM. Table 2 also shows the AUC of the teacher model on $D_{test}$ without and with taxonomy loss even though optimizing on taxonomy loss (equation 5) harms AUC, the model with taxonomy loss can handle product type errors and disambiguates homonym words based on context better than the model without taxonomy loss. Further, in the ablation study 4.3, we describe the importance of taxonomy loss for sourcing applications.

We also report Query-Query neighbourhood quality based on irrelevance on $D_{queries\_100K}$ to estimate the impact of taxonomy loss. In order to calculate this metric, first, we create a semantic index using $D_{queries\_100K}$ and for each query in $D_{queries\_100K}$ we get top-K (K=50) neighbours from the semantic space resulting in 5M <query, query> pairs (Note that, we fetched top-51 nearest queries and removed top-1 since this will be same as the original query). We report the neighbourhood quality metric by measuring the number of irrelevant queries pairs in the generated 5M <query,query> dataset. We classify two queries as irrelevant if they do not share any common browse nodes in the taxonomy tree. This 'no common browse node' approach is similar to the one used to generate $D_{QQ-}$. In Ablation Section 4.3.5, we explain why we cannot use the taxonomy tree to generate relevant pairs. From Table 2 we can observe although there is a drop in AUC on the <query, product> relevance task, there is a significant decrease in QQ-irrelevance with taxonomy loss.

**HISS** We also used $D_{test}$ dataset to measure the performance of hybrid inference architectures. Table 5 shows that both the variants of hybrid architectures perform better in terms of AUC than Student only architecture on the Query-Product relevance dataset. Table 4 shows few examples where hybrid architecture correctly captures the irrelevant <query,product> pairs where as the student only model fails to do so. Further we also report precision numbers of various inference architectures, for each query in $D_{queries\_100K}$ we retrieve K(50 and 100) nearest products in the semantic space of 5M products $D_{products\_5M}$. We measure the precison@K by measuring the number of relevant <query, product> pairs in this retrieved dataset. Table 5 shows that HISS performs better than Student-only model.

**Latency** We also measured the retrieval latency of BERT and DSSM models in online settings for embedding based sourcing applications. We leveraged HNSW library[22] for the indexing step (mlinks=64 and ef_construction=256). We report latency by taking the average retrieval time for queries in $D_{queries\_100K}$ using only CPU cores(on p3.8x machine); the reported latency includes feature engineering, embedding generation and KNN (k=100) retrieval. The results in Table 5 clearly demonstrates that BERT requires higher inference latency and hence requires more hardware to reach the same TPS (transactions per second) as that of the DSSM model. In many real-world applications, the BERT latency is not acceptable for online inference. Hence, there is a need for KD-DSSM and **HISS** inference architecture in online serving to achieve a good trade-off between latency and performance metric.

**Table 3: Effect of KD-loss terms**

| Model+loss | AUC |
|---|---|
| DSSM | 0.886 |
| KD-DSSM+ Alignment loss | 0.9033 |
| KD-DSSM + Interaction Loss | 0.9069 |
| KD-DSSM+ Alignment + Interaction Loss | 0.9038 |

**Table 4: Examples of <query,product> pairs correctly identified as negative by HISS(Query->KD-DSSM, product->SBERT) which the student-only model missed**

| Query | Product Title |
|---|---|
| hyundai stickers for car | AutoRetail Car Body Cover for Hyundai Eon(Silver Matty) |
| Ikea wooden sofa | HomeTown Bolton Queen Size Engineered Wood Bed With Box and side table |

## 4.3 Ablation studies

### 4.3.1 *How does taxonomy loss influence the quality of embedding space?*

We compared the top 10 neighbouring queries (based on cosine similarity) from the model output of SBERT trained with and without taxonomy loss for a set of randomly sampled queries. Below are some manually identified negatives of the model trained without taxonomy loss that get corrected after applying taxonomy loss : *(thread cutter scissors, nut cutters), (v11 cover, bag cover), (men handbag, hand purse for women), (gown for women, bathing clothes for women) and (puma shoes, puma backpack).* These are typical queries with homonym words with very different meanings that have to be derived from the context, and we use output of Query Classification model as a cue for disambiguation. This shows that the taxonomy loss is essential to control irrelevance.

### 4.3.2 *Isn't it better to jointly train the teacher and student model?*

The objective of the KD method is to allow the student model to learn parameters by imitating the teacher's prediction. The proposed approach can also be applied by allowing the student and teacher network to train jointly, instead of pretraining the teacher model and freezing the teacher weights during the KD process. Our experiments show that jointly training the teacher and student model hurts both teacher (AUC falls from 0.928 to 0.89) and student (AUC falls from 0.9038 to 0.881)) performance drastically.

### 4.3.3 *Do we require both alignment loss and interaction loss in our KD method?*

**Table 5: ROC-AUC, Latency and Precision@K(50,100) of various inference architecture**

| Inference Architecture | AUC | Latency | P@50 | P@100 |
|---|---|---|---|---|
| Teacher Model Only<br>Query→SBERT, ASIN→SBERT | 0.9283 | 18.46ms | 0.912 | 0.898 |
| Student Model Only<br>Query → KD-DSSM, ASIN→KD-DSSM | 0.9038 | 4.8ms | 0.903 | 0.8795 |
| Hybrid Architecure 1 : Direct Semantic Sourcing<br>Query→ KD-DSSM, ASIN→ SBERT | 0.9112 | 4.92ms | 0.9076 | 0.89 |
| Hybrid Architecure 2 : Keyword Recommendation<br>Query->SBERT, ASIN→ KD-DSSM | 0.9153 | 5ms | 0.905 | 0.886 |

Our experiments show that the model trained only with either alignment loss or interaction loss shows improvement on top of student model trained without KD. However, Table 3 shows that combing alignment and interaction loss dropped AUC by 0.34% when compared to the model trained only with interaction loss. Nonetheless, alignment loss is required for model interoperability. Section 3.5 describes the importance of model interoperability.

#### 4.3.4 *How do query lengths affect the performance of various models?*

We carried out query level analysis to understand the performance of KD-DSSM, DSSM and HISS architecture. Table 7 presents the performance of the models for varying query lengths. Here we observe that at lower query-length (1,2) student-KD-DSSM outperforms the teacher-SBERT model. This behaviour is intuitive as the probability of contextual words impacting semantics is less in queries of length 1 and 2. The study also demonstrates that the performance gap between SBERT and KD-DSSM increases as the query length increases. It is worth noting that the hybrid architecture is moderately resilient to this effect and hence bridges the performance gap between non-hybrid architectures and SBERT. Empirically, we show that KD-DSSM is suitable for lower query lengths(<3) but hybrid architectures is better suited for higher query lengths(>=3).

#### 4.3.5 *Why can't we use taxonomy tree to generate Query-Query positive pairs?*

As described in Section 3.2.3, E-commerce companies generally categorise products into a predefined multi-level product taxonomy. However, we cannot exploit this taxonomy tree to generate Query-Query or Product-Product relevant pairs because of the following two reasons (i) the nodes in the taxonomy tree are very coarse for many categories. For instance, in browse node: Health & Personal Care->Sports-Supplements->Gym Workouts (leaf node) will have varying products ranging from caffeine capsules to Amino-blend powders. So, showing muscle building powder products when the entered customer query is "caffeine capsules" leads to a terrible customer experience. (ii) Even when the leaf browse node is a fine-grained category for instance: Smartphones -> backcase, all the products within this category are not relevant to each other (Apple 8 backcase vs Apple 10xr backcase).

#### 4.3.6 *Why do we need to build a Query-Query similarity dataset using NPMI? Why can't we use a $D_{purchase}$ dataset directly?*

Queries are usually shorter in length compared to product title and generally not polluted by superfluous words. Hence, by leveraging the Query-Query dataset, the model learns better embeddings by learning to concentrate only on the relevant tokens. Our experiments also showed that our model could correctly retrieve semantically similar products even when the queries are expressed in vernacular terms by leveraging the Query-Query dataset. In contrast, when we leveraged the same query-product (purchased) directly, our model failed to retrieve them. Table 6 shows some <query,product> pairs which are retrieved after leveraging $D_{QQ+}$ dataset.

**Table 6: Examples of <query,product> pairs retrieved after leveraging $D_{QQ+}$ dataset**

| Query | Product Title |
|---|---|
| payal | Shining Diva Fashion Oxidised Silver Floral Single Stylish Anklet for Women & Girls |
| geezer | Crompton Arno Neo 10-L 5 Star Rated Storage Water Heater with Advanced safety |
| omedone extractor | Havells SC5060 Pore Cleanser, Blackhead/ Whitehead Remover, 3 Suction Modes- Low/Medium/High Fast Charge (White) |

## 5 CONCLUSION

In this paper, we applied KD to distil information from SBERT to DSSM, and showed that the performance of DSSM model is comparable to SBERT. Further, we proposed a novel alignment loss function in KD method to enable model interoperability. Through experiments we showed the importance and applications of enabling model interoperability to build **HISS**-hybrid inference architecture such that we can leverage the best of SBERT-high capacity model and DSSM-low latency model in real-time inference pipeline.

## REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang, and Jimmy Lin. Cross-domain modeling of sentence-level evidence for document retrieval. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3490–3496, Hong Kong, China, November 2019. Association for Computational Linguistics.

[3] Junjie Bai, Fang Lu, Ke Zhang, et al. Onnx: Open neural network exchange. https://github.com/onnx/onnx, 2019.

[4] Michael Bendersky, Donald Metzler, and W. Bruce Croft. Learning concept importance using a weighted dependence model. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, page 31–40, New York, NY, USA, 2010. Association for Computing Machinery.

[5] Hang Cui, Ji-Rong Wen, Jian-Yun Nie, and Wei-Ying Ma. Probabilistic query expansion using query logs. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, page 325–332, New York, NY, USA, 2002. Association for Computing Machinery.

[6] Zhuyun Dai and Jamie Callan. Deeper text understanding for ir with contextual neural language modeling. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Jul 2019.

[7] Zihang Dai, Guokun Lai, Yiming Yang, and Quoc V. Le. Funnel-transformer: Filtering out sequential redundancy for efficient language processing, 2020.

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

[9] Iftah Gamzu, Marina Haikin, and Nissim Halabi. Query rewriting for voice shopping null queries. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1369–1378, New York, NY, USA, 2020. Association for Computing Machinery.

[10] Saurabh Goyal, Anamitra R. Choudhury, Saurabh M. Raje, Venkatesan T. Chakaravarthy, Yogish Sabharwal, and Ashish Verma. Power-bert: Accelerating bert inference via progressive word-vector elimination, 2020.

[11] Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, and Narayan Bhamidipati. Context- and content-aware embeddings for query rewriting in sponsored search. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 383–392, New York, NY, USA, 2015. Association for Computing Machinery.

[12] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[13] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. Efficiently teaching an effective dense retriever with balanced topic aware sampling, 2021.

[14] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information amp; Knowledge Management*, CIKM '13, page 2333–2338, New York, NY, USA, 2013. Association for Computing Machinery.

[15] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding, 2020.

[16] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert, 2020.

[17] Giridhar Kumaran and Vitor R. Carvalho. Reducing long queries using query quality predictors. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, page 564–571, New York, NY, USA, 2009. Association for Computing Machinery.

[18] Jey Han Lau, David Newman, and Timothy Baldwin. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 530–539, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

[19] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-yi Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 912–921, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[20] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[21] Wenhao Lu, Jian Jiao, and Ruofei Zhang. Twinbert: Distilling knowledge to twin-structured bert models for efficient retrieval, 2020.

[22] Yu. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs, 2018.

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

[24] Akash Kumar Mohankumar, Nikit Begwani, and Amit Singh. Diversity driven query rewriting in search advertising, 2021.

[25] Priyanka Nigam, Yiwei Song, Vijai Mohan, Vihan Lakshman, Weitian (Allen) Ding, Ankit Shingavi, Choon Hui Teo, Hao Gu, and Bing Yin. Semantic product search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery  Data Mining*, KDD '19, page 2876–2885, New York, NY, USA, 2019. Association for Computing Machinery.

[26] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[27] Stefan Riezler and Yi Liu. Query rewriting using monolingual statistical machine translation. *Comput. Linguist.*, 36(3):569–582, September 2010.

[28] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

[29] Dou Shen, Ying Li, Xiao Li, and Denny Zhou. Product query classification. In *CIKM '09 Proceedings of the 18th ACM conference on Information and knowledge management*, pages 741–750. ACM Press, November 2009.

[30] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Gregoire Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *CIKM*, November 2014.

[31] Michael Skinner and Surya Kallumadi. E-commerce query classification using product taxonomy mapping: A transfer learning approach. In *eCOM@SIGIR*, 2019.

[32] Alessandro Sordoni, Yoshua Bengio, and Jian-Yun Nie. Learning concept embeddings for query expansion by quantum entropy minimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 28(1), Jun. 2014.

[33] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020.

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

[35] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

[36] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. Deebert: Dynamic early exiting for accelerating bert inference, 2020.

[37] Jinxi Xu and W. Bruce Croft. Quary expansion using local and global document analysis. *SIGIR Forum*, 51(2):168–175, August 2017.

[38] Jing Yang, Brais Martinez, Adrian Bulat, and Georgios Tzimiropoulos. Knowledge distillation via softmax regression representation learning. In *International Conference on Learning Representations*, 2021.

[39] Jing Yang, Brais Martinez, Adrian Bulat, and Georgios Tzimiropoulos. Knowledge distillation via softmax regression representation learning. In *International Conference on Learning Representations*, 2021.

[40] Han Zhang Songlin Wang Sulong Xu Yun Xiao Bo Long Wen-Yun Yang Yiming Qiu, Kang Zhang. Query rewriting via cycle-consistent translation for e-commerce search, 2021.

[41] Zhi Zheng, Kai Hui, Ben He, Xianpei Han, Le Sun, and Andrew Yates. Bert-qe: Contextualized query expansion for document re-ranking, 2020.

[42] Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit, 2020.


# A APPENDIX

*A.0.1* ***How do query lengths affect the performance of various models?*** We carried out query level analysis to understand the performance of KD-DSSM, DSSM and HISS architecture. Table 7 presents the performance of the models for varying query lengths. Here we observe that at lower query-length (1,2) student-KD-DSSM outperforms the teacher-SBERT model. This behaviour is intuitive as the probability of contextual words impacting semantics is less in queries of length 1 and 2. The study also demonstrates that the performance gap between SBERT and KD-DSSM increases as the query length increases. It is worth noting that the hybrid architecture is moderately resilient to this effect and hence bridges the performance gap between non-hybrid architectures and SBERT. Empirically, we show that KD-DSSM is suitable for lower query lengths(<3) but hybrid architectures is better suited for higher query lengths(>=3).

*A.0.2* ***Effect of alignment loss.*** Figure 4(Left) shows that SBERT and KD-DSSM (trained with interaction loss only) project the entities into their respective clusters in semantic space. Therefore embeddings generated for the same entity are not close to each other. However, Figure 4(Right) shows that there are no model-wise clusters formed when we train KD-DSSM with alignment loss. As a

**Table 7: ROC-AUC of Query-Product relevance task at varying levels of query length. The best performing model among SBERT, DSSM, KD-DSSM and hybrid inference architectures for each query length is highlighted.**

| Number of tokens in Query | SBERT | KD-DSSM | DSSM | Query ->SBERT Ad ->DSSM | Query ->DSSM Ad ->SBERT |
|---|---|---|---|---|---|
| **1** | 0.9258 | *0.9285* | 0.9264 | **0.9287** | 0.9227 |
| **2** | **0.9292** | *0.9223* | 0.9109 | 0.9202 | 0.9204 |
| **3** | **0.932** | 0.9065 | 0.8894 | *0.9177* | 0.9137 |
| **4** | **0.9284** | 0.9096 | 0.8789 | 0.9069 | *0.9194* |
| **5** | **0.9325** | 0.9088 | 0.871 | 0.9146 | *0.9202* |
| **6** | **0.9204** | 0.8894 | 0.8545 | *0.9057* | 0.9018 |
| **7** | **0.9228** | 0.8735 | 0.8575 | 0.8922 | *0.8937* |
| **8+** | **0.9052** | 0.8654 | 0.8462 | *0.8994* | 0.8944 |

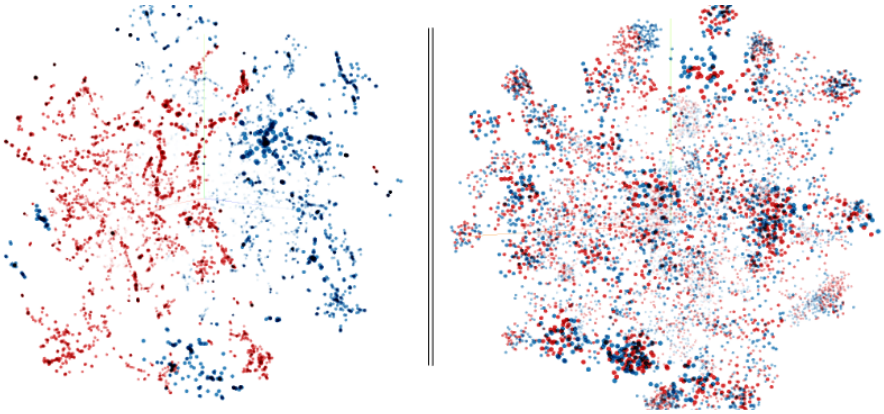

**Figure 4: T-SNE plots using embeddings from both SBERT and KD-DSSM generated for 4000 entities. Blue indicates SBERT embeddings and Red indicates KD-DSSM embeddings. Left : KD-DSSM without alignment loss and Right: KD-DSSM with alignment loss**

result, embeddings generated for the same entity are closer to each other, thereby illustrating model interoperability.