Beyond Hard Negatives in Product Search: Semantic Matching using One-Class Classification (SMOCC)

Arindam Bhattacharya*
Ankit Gandhi*
Amazon
Bangalore, India
aribhat@amazon.com
ganankit@amazon.com

Vijay Huddar
Amazon
Bangalore, India
vhhuddar@amazon.com

Ankith MS
Amazon
Bangalore, India
ankiths@amazon.com

Aayush Moroney
Amazon
Bangalore, India
amoroney@amazon.com

Atul Saroop
Amazon
Bangalore, India
asaroop@amazon.com

Rahul Bhagat
Amazon
Seattle, USA
rbhagat@amazon.com

ABSTRACT

Semantic matching is an important component of a product search pipeline. Its goal is to capture the semantic intent of the search query as opposed to the syntactic matching performed by a lexical matching system. A semantic matching model captures relationships like synonyms, and also captures common behavioral patterns to retrieve relevant results by generalizing from purchase data. Semantic matching models however suffer from lack of availability of informative negative examples for model training. Various methods have been proposed in the past to address this issue based upon hard-negative mining and contrastive learning.

In this work, we propose a novel method for semantic matching based on one-class classification called SMOCC. Given a query and a relevant product, SMOCC generates the representation of an informative negative which is then used to train the model. Our method is based on the idea of generating negatives by using adversarial search in the neighborhood of the positive examples. We also propose a novel approach for selecting the radius to generate adversarial negative products around queries based on the model's understanding of the query. Depending on how we select the radius, we propose two variants of our method: SMOCC-QS, that quantizes the queries using their specificity, and SMOCC-EM, that uses expectation-maximization paradigm to iteratively learn the best radius. We show that our method outperforms the state-of-the-art hard negative mining approaches by increasing the purchase recall by 3 percentage points, and improving the percentage of exacts retrieved by up to 5 percentage points while reducing irrelevant results by 1.8 percentage points.

*Both authors contributed equally to this research.

WSDM '23, February 27-March 3, 2023, Singapore, Singapore

ACM ISBN 978-1-4503-9407-9/23/02.
https://doi.org/10.1145/3539597.3570488

CCS CONCEPTS

• **Information systems** → **Information retrieval**; **Search and retrival**; • **Applied computing** → **Online shopping**.

KEYWORDS

semantic search, one class classification

ACM Reference Format:
Arindam Bhattacharya, Ankit Gandhi, Vijay Huddar, Ankith MS, Aayush Moroney, Atul Saroop, and Rahul Bhagat. 2023. Beyond Hard Negatives in Product Search: Semantic Matching using One-Class Classification (SMOCC). In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM '23), February 27-March 3, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3539597.3570488

1 INTRODUCTION

Modern Search Engines combine benefits from matches obtained through lexical, behavioral and semantic sources. The lexical matches allow the product search engine to scale to potentially 100s of millions of products, all within the constraints of a realtime search. The lexical matches are in particular useful when the vocabulary used to describe the product is the same as that used by the customer to search for it. The behavioral matches help the search system "memorize" the essential associations its customers have already taught it through their behavior. Through behavioral matches, the search engine memorizes that customers buy a "Samsung Galaxy Flip3" phone when searching for a "flip", but a "highend memory foam mattress" when they search for "restful sleep". On the other hand, semantic matches aid the search systems generalize and generate relevant matches even when the vocabulary of the product description does not overlap with that of the customer's query terms. Such matches help the search engine realize that "baking moulds", for example, qualify for the search query "mother's day gifts".

In this work, we concern ourselves with generation of semantic matches for product search engines. Typical approach to training such semantic search models is to use the historical click/add-to-cart/purchase inputs from customers, albeit in an impersonal manner, and use those to learn two-tower language models, typically with a shared vocabulary across the two towers [23]. Two-tower

based architectures are very popular in literature and practice to build semantic similarity models in product search. These models formulate the matching problem as a metric learning task where semantic embeddings of queries and products are learnt in a joint embedding space and approximate nearest neighbor based lookups are then performed to find semantically similar products to a query in realtime [13, 15, 18, 23, 39]. However, while the positive training data is available in abundance in such a setting, in terms of the customer behavior, high-quality negatives are hard to come by. Hence, multiple works [4, 11, 18, 36] have approached the problem through use of contrastive learning, while explicitly trying to find high-quality "hard negative" examples that could improve the quality of the semantic matching model. These works have inferred that the quality of the model, measured in terms of the percentage of exact matches in its output, improves with better quality hard negatives. However, in this work, we ask ourselves the question if the need for generating hard negative samples in semantic search models can completely be side-stepped instead by modeling the problem as a one-class classification problem.

This work contributes to the literature in the following manner:

(1) We propose a one-class classification approach to building semantic matching models for product search. In our understanding, this work represents the first reported effort with the approach.
(2) While the approach we take for one-class classification is heavily influenced by [9], we carry out necessary modifications to the method to expand its applicability to language models. In particular, we realize that, unlike image datasets explored in [9], perturbations to the positive samples in the input space do not yield negative samples for language-related applications. We instead carry out such perturbations to intermediate representations to generate negative samples to train on.
(3) Most previous works of semantic matching, including [9] operate under the assumption of a fixed radius or high-dimensional annular around positive examples. Irrespective of the characteristics of the query, positive product examples are embedded within this radial or annular structure and negative examples are forced out. In this work, we posit that the size of such a radial or annular structure should be a function of the characteristics of the query issued by the customers. In particular, if a query is highly specific, the size of the annular structure should be small, while if the query is generic in nature, then the size of the annular structure can be relatively larger.
(4) Even when we build localized query-specific loss structures around positive samples, there are two potential approaches. One approach is to force-fit a query characteristic dependent yet pre-decided rigid loss structure around the positive examples. In such an approach, introduced as *SMOCC-QS* later in the paper, we hope to use the flexibility associated with high-dimensional training of query and product embeddings to adapt itself to the pre-decided rigid loss structure. The second approach, introduced as *SMOCC-EM*, which we find to be superior, is to parameterize the query characteristic dependent loss structure and learn its parameters jointly while learning the semantic matching model.

This paper is organized as follows. Section 2 discusses previous work carried out in literature, while Section 3 introduces our proposed method. Section 4 provides the details of the experiments we have carried out to evaluate the relative performance of various proposed methods against current state-of-the-art baselines. Sections 4.4 and 5 discuss the results and conclude the paper.

2 BACKGROUND AND RELATED WORKS

Product Search. With advancements in Natural Language Processing (NLP), various search engine systems have adopted semantic based retrieval in their pipelines [6, 12, 18]. Word2Vec [21] revolutionized NLP by representing words as dense vectors, so that two semantically similar words would have similar representations. As a result, many studies have been conducted in semantic matching to represent queries and documents as dense vectors and use them for document retrieval. Some related works with this approach are DSSM [13], CDSSM [33] and Product Semantic Search [23]. Semantic matching for products has also been explored using extreme multilabel classification systems. These algorithms use sparse TF-IDF features on textual inputs and leverage different partitioning techniques on the label space to reduce complexity [2].

In the last couple of years, transformers [35] and large language models like BERT [5] and RoBERTa [17] have achieved state-of-the-art in various NLP tasks, and transformer-based models have emerged as the de-facto standard for generating text-based embeddings. These language models learn high quality query and document representations using a self-attention mechanism - every input word interacts with every other input word to produce a contextualised representation. Furthermore, the model is pretrained by generating labeled training data from massive amounts of unsupervised data, which helps achieve good performance with less data [16]. While such embedding generation models have advanced, they still require a considerable number of positive and negative pairs to build an effective production-level product matching model to ensure only relevant products are sourced for queries [16].

Hard Negative Mining. Traditionally, in order to train semantic matching models, methods have needed both positive and negative <query, product> pairs to be obtained. Historical anonymized customer session logs containing queries, clicks and purchase data are leveraged to generate relevant samples [23]. For example, if many customers search for the term "shoes" but ultimately in the same session end up purchasing "sneakers", then the model learns that "shoes" and "sneakers" are relevant to each other. Negative pairs are usually generated randomly. But, [30] showed that model trained using only random negatives places two dis-similar queries closer to each other in the embedding space, especially when such queries have shared tokens. For instance, queries such as "thread cutter scissors" and "nut cutters". Also, [38] showed that semantic matching models trained on hard negative samples perform better than those trained with randomly selected negatives. While hard negative mining is popular, much of the literature is concentrated in the field of computer vision [10, 37] and classification application [32]. This makes it hard to apply the existing techniques for the search retrieval application which operates on a discrete domain and also does not have a concept of "classes".

Contrastive learning is one of the common techniques to mine difficult negatives in machine learning. Recently, it has been explored in NLP to learn sentence embedding [3, 7, 40]. The disadvantage of contrastive learning methods for negative sampling is that they have a low probability of obtaining any significant negatives from random samples, thereby limiting the number of meaningful negatives one can obtain. For obtaining more negatives, [4, 10] keep a queue of features from recent batches as a memory bank. [14] built upon [10] by generating hard negative examples through mixing positive and negative examples in the memory bank. [30] used product taxonomy information to generate hard negatives.

One Class Classification. One-class classification deals with training classification models in the absence of more than one class labels. There exists a wide variety of approaches towards one-class classification [29], which are widely divided into SVM based approaches, projection based approaches, and deep neural network based approaches [9, 28]. SVM based approaches formulate the one-class classification problems as a constrained optimization by either creating a hyper-sphere to contain the normal points [31], or to learn a hyper-plane separating the origin from the normal points [34]. Projection based approaches use random projections [1], histogram based binning [8] or density estimation [25] to reduce the dimension of the data, and uses some form of clustering to obtain the decision boundaries. Deep neural network based approaches either learn representation of the normal points and use SVM based approach to determine the boundaries [28], perform adversarial training [24], or generating negative examples using optimization [9]. We adopt the approach mentioned in [9] and adapt it for semantic matching task.

3 PROPOSED METHOD

In this section, we present the proposed approach for semantic matching using one-class classification (SMOCC). Figure 1 shows the overall architectural diagram for the proposed method. In semantic product matching, given a input query q_i, the task is to retrieve all the relevant products p_i from the catalog within the latency constraints of a realtime search. As noted in Section 1, most approaches have tried to solve the problem of building accurate semantic matching models through methods that find high quality negative examples for the models to train on. In this work, we use one-class classification based formulation to adversarially generate hard negative products for a query and train semantic matching models. SMOCC is based on the hypothesis that the set of relevant products for a query lie on a locally linear low-dimensional manifold, within the high-dimensional embedding space in which the queries and products are embedded. Under this assumption, as manifolds are locally Euclidean in nature, we can use the standard l_2 distance function to compare the embeddings of products within a small neighborhood of a query embedding. Typically, in such a setting, majority of the points outside a small radius are found to be irrelevant to the query. We use query specific radius values to ensure that the generated products are 'hard' for a query. For e.g.: for generic queries like 'men's shirts' and 'phone under $500', we choose high radius values as compared to radius for specific queries such as 'arrow black printed shirt for men' and 'iphone pro 12'. This information allows us to synthetically generate negative products while training via a gradient ascent phase similar to adversarial training [19]. Our method is based on DROCC [9] and it uses gradient ascent phase to adaptively add generated irrelevant query-product pairs to our training set, and a gradient descent phase to minimize the classification loss by learning a representation and a classifier to separate relevant query-product pairs from generated irrelevant ones.

3.1 DROCC overview

Deep robust one-class classification (DROCC) [9] is based on a low-dimensional manifold assumption on the positive class for anomaly detection. It synthetically and adaptively generates negative examples to provide a robust approach for one-class classification. Let $S = \{x_1, x_2, \dots x_n\}$ denote the set of non-anomalous points, then DROCC learns a function $f_\theta : \mathbb{R} \to \{-1, 1\}$ such that $f_\theta(x) = 1$ when $x \in S$ and $f_\theta(x) = -1$ when $x \notin S$. The anomaly classification model f is parametrized by parameters θ. Mathematically, DROCC optimizes the following loss function -

$$\min_\theta \left\{ \lambda \|\theta\|^2 + \sum_{i=1}^{n} L(f_\theta(x_i), 1) + \mu \max_{\tilde{x}_i \in N_i(r)} L(f_\theta(\tilde{x}_i), -1) \right\} \quad (1)$$
$$N_i(r) \stackrel{\text{def}}{=} r \le \|\tilde{x}_i - x_i\|^2 \le \gamma . r$$

where, λ, μ are regularization parameters; L denotes the loss function such as cross-entropy loss; $N_i(r)$ denotes the points that are not on the manifold, and are at least r distance away from a given point and upper bounded by $\gamma \cdot r; \gamma \ge 1$. The objective here is to classify the given non-anomalous x_i's as positives and generated anomalous points $\tilde{x}_i$'s as negatives. DROCC first computes the loss of the network w.r.t a negative label and then maximizes this loss in order to find the most effective adversarial negative point via gradient ascent. The generated negative point is used along with the given positive point to minimize the classification loss.

In this work, we use a two-tower bi-encoder model inspired from [23] to learn query and product representations (see Figure 1) consisting of embedding, pooling, normalization and a couple of fully-connected dense layers. We also experiment with BERT based bi-encoder models and show results in Section 4.7. However, deploying BERT-based models in online scenarios is challenging because of latency constraints.

3.2 Obtaining Embeddings in Low-dimensional Manifold

DROCC [9] is motivated by the key observation that generally, the typical data of interest lies within a low-dimensional manifold. It works by perturbing a normal input point in the input space by a certain distance. The perturbation is done by a random vector with a certain norm, called the radius. This kind of perturbation works only if the input space is Euclidean and the non-anomalous points lie within a low dimensional manifold of the input space. This hypothesis holds true for many domains such as images and speech where input pixels or signals can be perturbed. However, in the natural language domain used for semantic matching, this is not straightforward. To address this, in SMOCC, we perform a pre-training phase to obtain embeddings of text that satisfy this criteria. First, we define few notations. Let

$\mathcal{S} = \{q_i, p_i\}$ denote the set of relevant query-product pairs. Using SMOCC, we generate adversarial negatives $\tilde{p}$, and learn functions $f_{\theta_1}, f_{\theta_2}$ to get representations of queries and products respectively such that the similarity of relevant query-product pair is higher than the similarity of irrelevant query-product pair, i.e., $\|f_{\theta_1}(q) - f_{\theta_2}(p)\|^2 < \|f_{\theta_1}(q) - f_{\theta_2}(\tilde{p})\|^2, \forall p, q, \tilde{p}$.

For this phase, we generate m negative examples per query by randomly shuffling the products corresponding to the queries. This gives us the set of negative examples $\mathcal{N} = \{(q_i, p_j)\}_{i,j\in 1, i\neq j}^{n\times m}$. We use $\mathcal{S} \cup \mathcal{N}$ to pre-train the two-tower bi-encoder model using an Euclidean distance based similarity measure defined as

$$sim(p, q) = 1 - \tanh(\|f_{\theta_1}(q) - f_{\theta_2}(p)\|^2) \tag{2}$$

where $f_{\theta_1}(q)$ and $f_{\theta_2}(p)$ are query and product embeddings, respectively. This similarity score is used with MSE loss to train the two-tower model in SMOCC in the pre-training phase. Because of the Euclidean distance based similarity measure, the resultant embeddings of queries and products tend to confirm to the hypothesis that the input points should lie in the embedding space. Specifically, for semantic matching, given a pair $(p, q) \in \mathcal{S}$, the intermediate representation $f^l_{\theta_1}(q)$ and $f^l_{\theta_2}(p)$ from layer l of the network are such that perturbing $f^l_{\theta_2}(p)$ by a distance r results in a pair $(q, \hat{p})$ such that $(q, \hat{p}) \notin \mathcal{S}$ (refer to architectural diagram in Figure 1). Note that the architecture may be modified to share the parameters of query and product embeddings.

3.3 SMOCC formulation

Mathematically, we optimize the following optimization problem in SMOCC -

$$\begin{aligned}
&\min_{\theta_1,\theta_2} \left\{ \alpha\|\theta_1\|^2 + \beta\|\theta_2\|^2 + \sum_{i=1}^{n} L\left(f_{\theta_1}(q_i), f_{\theta_2}(p_i), f_{\theta_2}(\tilde{p}_i)\right) \right\} \\
&\tilde{p}_i = \max_{\tilde{p}_i \in N_i(r_i)} L\left(f^l_{\theta_1}(q_i), f^l_{\theta_2}(p_i), f^l_{\theta_2}(\tilde{p}_i)\right) \text{ ,where ,} \\
&L(p, q, \tilde{p}) = \log(1 + e^{(\|f_{\theta_1}(q) - f_{\theta_2}(p)\|^2 - \|f_{\theta_1}(q) - f_{\theta_2}(\tilde{p})\|^2)}) \\
&N_i(r_i) \stackrel{\text{def}}{=} r_i \leq \|f_{\theta_1}(q_i) - f_{\theta_2}(\tilde{p}_i)\|^2 \leq r_i + \gamma
\end{aligned} \tag{3}$$

α, β are regularization parameters; θ_1 and θ_2 are the learnable parameters of query and product arms respectively in the bi-encoder model, L denotes the triplet loss function between positive and negative query-product pairs; $N_i(r_i)$ records products off the manifold of relevant products for a query q_i and are between distance r_i and $(r_i + \gamma)$ from the query; $\tilde{p}_i$ denotes an adversarial negative product for the query q_i; $f^l_{\theta_1}$ and $f^l_{\theta_2}$ are outputs corresponding to intermediate layer of the bi-encoder network.

The goal of SMOCC is to rank relevant query-product pairs $(q_i, p_i), i \in \mathcal{S}$ higher than the generated adversarial query-product pairs $(q_i, \tilde{p_i})$. SMOCC achieves this by minimizing a ranking loss using triplet loss between $\{p_i, q_i, \tilde{p}_i\}$. The adversarial products are generated for a query by maximizing the triplet loss using the intermediate representations of queries and products from the bi-encoder network via gradient ascent. In the next few subsections, we describe how the radius values r_i are calculated for different queries.

3.4 Radius selection for queries

Selection of right radius values (r_i) for queries is extremely important in SMOCC. Fixing a single radius value for all queries often lead to suboptimal performance because the specificity levels of queries are not same. We define query specificity is a measure of how specific is the intent of the query (or conversely, how broad is the intent). For example, a query such as 'shoes' has a broad intent with a wide variety of products satisfying the intent of the query whereas a query such as 'black docker leather shoes' has a specific intent with fewer products satisfying the query intent. Hence, in order to generate accurate hard negatives for a query, the radius values have to be set appropriately based upon the specificity levels of the query, i.e., broader the intent of the query, higher the radius. As part of SMOCC, we have proposed two novel radius selection algorithms (described in below subsections), one based upon estimation of query specificity levels of queries, called *SMOCC-QS* and another based upon expectation-maximization (EM) framework, called *SMOCC-EM*, to alternatively predict radius values and learn the semantic matching model. For both the methods, we initialize the radius values based on the embedding distances after the warm-up steps. As the model learns better embeddings, the choice of radii improves for both the proposed methods.

3.5 Radius selection using query specificity estimation

We estimate query specificity empirically using historical behavioral search logs. We use the distribution of clicks over products for a query to compute its specificity. Queries with specific intent receive clicks on fewer products than queries with broad intent. As a result, the distribution of clicks is less spread in specific queries compared to broad queries. To characterize the spread of a probability distribution, we use a common statistical measure – information entropy. We use last 1 year search logs to aggregate the total count of clicks per product for a query. We use this aggregated data to generate the probability distribution of clicks over all products for a query. We compute the information entropy of the probability distribution to get the click-entropy score for the query. The negative of click-entropy score represents the query specificity (*QS*).

$$QS = \sum_i P_i \cdot \log(P_i) \tag{4}$$

where P represents the probability distribution of clicks over products. We compute QS score for all queries in the training dataset and divide the queries into Q_0 bins based upon their specificity scores. For each bin, we select a radius value proportional to average distance between embeddings of positive query-product pairs in that bin. We refer to this method as *SMOCC-QS* and present the complete SMOCC algorithm for this approach in Algorithm 1.

We estimate query specificity from historical behavior information about the query. One drawback of the above approach is that it will not estimate specificity score for unseen queries or very low frequency queries with insufficient historical information. However, anecdotally we have seen that most such cases are very specific queries with customers carrying out highly focussed searches.

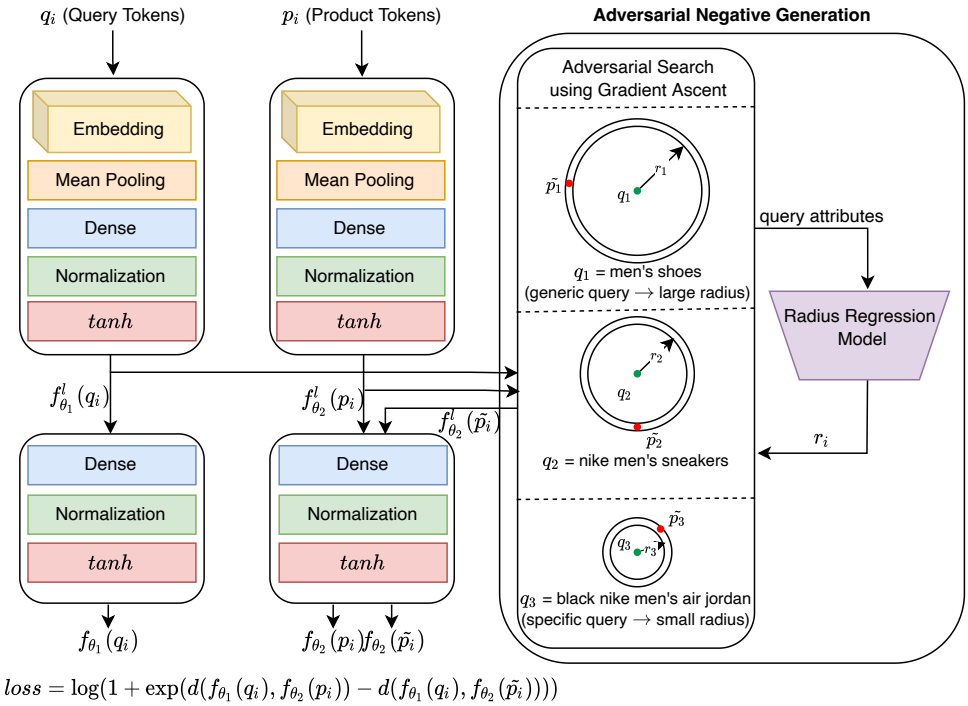


Figure 1: Architecture of SMOCC. Given the input query q_i and product p_i, the two tower architecture generates an intermediate representation $f^l_{\theta_1}(q_i)$ and $f^l_{\theta_2}(p_i)$. The negative samples are generated using the intermediate representation using adversarial search with a radius value determined by the regression model. The regression model decides the radius value based on query attributes. Generic queries have larger radius. As the query specificity increases, the radius decreases as shown in the three examples in the figure. Finally, the two tower model is trained using the positives and generated negatives using a triplet loss.

3.6 Radius selection using EM framework

In this section, we propose an Expectation-Maximization (EM) based framework to alternatively estimate radius values for queries and train SMOCC model using them. As SMOCC-QS cannot estimate radius values for unseen or low frequency queries, in this approach, we build a tree-based regressor to estimate radius values for all kinds of queries. Also, the regressor here utilizes other query signals such as product type in the query, order rate of the query, percentage of the times query has been reformulated, etc. along with query specificity scores to estimate the radius for query. The target values for the regressor are determined from historical behavior search logs. For a query, we find out all products that are relevant to it in $\mathcal{S}$ and take the average value between the distance of the query and the relevant products as the target of the regressor for that query. Firstly, we initialize the SMOCC model with a Euclidean distance based MSE loss (Equation 2). Secondly, we train the regressor model to estimate radius values by defining the target values from the learned embeddings. We alternate between these two steps for e_0 number of steps until the bi-encoder model starts over-fitting or radius values for queries converges. We refer to this approach as *SMOCC-EM* and present the algorithm in Algorithm 2. Each iteration of the M step is similar to an iteration of SMOCC-QS. The overhead added by the E step is small because the model used is a simple tree based regressor with few features.

3.7 Curriculum Learning based training of SMOCC

We train both SMOCC-QS and SMOCC-EM models with a curriculum learning training strategy where a model is trained from easier data to harder data. More specifically, the basic idea is to first train SMOCC models with easier queries (queries with high radius values), and then gradually increase the difficulty level of queries. Queries with high radius values often have broader intent, and it is easier to retrieve exact product matches for them at the top than the very specific queries because of low number of exacts for them in the catalog. In SMOCC-QS, we train model from higher score bins to lower score bins. And in SMOCC-EM, we sort the queries based upon descending values of their radii and pass them to M step in that particular order. Curriculum learning helps us in improving the generalization capacity of the SMOCC models.

4 EXPERIMENTS

We now compare the performance of the SMOCC methods with the state-of-the-art baselines for semantic matching in product search. In the rest of the section, we discuss the dataset, experimental setup, the evaluation criteria and the results. We also discuss the effects of curriculum learning and the sensitivity to various parameters in both SMOCC methods.

Algorithm 1: Semantic matching model training via SMOCC-QS

Input Dataset: Relevant query-product pairs $D = \{q_i, p_i\}_{i=1}^{n}$
Learnable Parameters: θ_1, θ_2, r_i
Pre-training: For b_0 batches,
(1) Add positive pairs (q_i, p_i) in the batch.
(2) For every query q_i, add m random negatives ((q_i, p_j)'s, $j \neq i$) in the batch.
(3) Compute loss based upon MSE distance (Equation 2)
(4) Backpropagate loss and update weights θ_1 and θ_2 in the network.
Radius Estimation for queries:
(1) Divide queries (q_i's) into Q_0 bins based upon their QS scores (Equation 4).
(2) For each bin $b \in Q_0$, radius is $avg_i(\|f_{\theta_1}(q_i) - f_{\theta_2}(p_i)\|^2), \forall i \in b)$.
(3) Query radius r_i = radius of bin to which it is assigned.
One-class classification training: For b_0' batches,
(1) Get $\tilde{p}_i$ by performing adversarial search using gradient ascent $\forall q_i$ in the batch. It is done by taking intermediate representations from layer l in the bi-encoder network (Equation 3).
(2) Compute triplet loss between q_i, p_i and $\tilde{p}_i$.
(3) Backpropagate loss and update weights θ_1 and θ_2 in the network.

Algorithm 2: Semantic matching model training via SMOCC-EM

Input Dataset: Relevant query-product pairs $D = \{q_i, p_i\}_{i=1}^{n}$
Learnable Parameters: θ_1, θ_2, r_i
Pre-training: Same as SMOCC-QS
EM algorithm: *While the model does not overfit or radius values r_i do not converge:*
- **E step - radius estimation for queries:**
  (1) Get query attributes such as query product type, query specificity scores, etc. to build a regressor.
  (2) Set the target values in regressor based upon M step. For a query q_i, retrieve all p_i's such that $(q_i, p_i) \in \mathcal{S}$ and set the target to $avg_i(\|f_{\theta_1}(q_i) - f_{\theta_2}(p_i)\|^2)$.
  (3) Train regressor and $\forall q_i$, predict r_i.
- **M step - one-class classification training:** For b_0' batches,
  (1) Get $\tilde{p}_i$ by performing adversarial search using gradient ascent for $\forall q_i$ in the batch with r_i from E-step. It is done by taking intermediate representations from layer l in the bi-encoder network (Equation 3).
  (2) Compute triplet loss between q_i, p_i and $\tilde{p}_i$.
  (3) Backpropagate loss and update weights θ_1 and θ_2 in the network.

4.1 Dataset and Experimental Setup

We perform our experiments on the historical purchase data from a popular eCommerce product site. We randomly sample one year of purchases and the corresponding customer search queries from the marketplace, of which we use the first 10 months as training data and one month of data each for model validation and testing respectively. Using above random sampling, we create training and validation datasets containing 3 million and 1 million relevant query-product pairs respectively. For model evaluation, we create a test dataset containing 10K queries and 205K products randomly sampled from the testing period. We evaluate the queries in test dataset in a retrieval setting.

In addition, we also the KDDCup 2022 ESCI challenge dataset provided by Amazon [26]. We filter the data on *US* locale. The dataset contains 1.25 million query-product pairs each labelled with ESCI labels. We split the dataset into training, validation and test set consisting 80%, 10% and 10% of the data, respectively.

We use query keyword and title of the product as input in all our baselines and proposed models. We fix the embedding sizes of queries and products to 256 in all the methods. We use a batch size of 256. Note that unlike some baselines where we mine hard negatives in batch, SMOCC is not sensitive to the batch size. The methods were implemented using PyTorch on a machine running on 64 core Intel Xeon CPU @ 2.30GHz 16 Tesla K80 GPUs with 11 GB memory. We train the SMOCC models using AdamW optimizer with exponential learning rate decay with initial learning rate of 0.05. All the models are trained for 40 epochs and pre-trained for 10 epochs. For SMOCC-QS, we train the model by dividing the query specificity into Q_0=5 bins. Later in the section, we show the effect of changing the number of bins. α, β, and γ are set to 0.01, 0.01 and 1 respectively. Random forest regressor is used as regressor in SMOCC-EM. In SMOCC-EM, each M step constitute 10 epochs of training of bi-encoder model, i.e., after initial 10 epochs of pre-training, we alternate the *E*-step of training the regression model using query attributes, and *M*-step of training the one-class classification model for 10 epochs.

4.2 Evaluation metrics

We obtain the learned representation of queries and products for models under contention. Then, a KNN index using approximate nearest neighbor method (HNSW [20]) is built for 206K products from the test dataset. To evaluate the efficacy of our approach, we retrieve the top-k semantically similar products for a given search query in the index and evaluate them on following metrics -

- **ESCI% in top-k** [27]: This metric is used to evaluate the relevancy of results returned by the model. We fix $k = 100$ in all our experiments for internal dataset, and $k = 5$ for public dataset (due to lack of sufficient samples for higher k), and compute percentage of exacts, substitutes, complements and irrelevants in top-k. Exact means product is relevant to the query; substitute is somewhat relevant to the query; complement does not fulfill the query, but could be used in combination with an exact product corresponding to the query; and irrelevant fails to fulfill a central aspect of the query. We evaluate all the models using our in-house deep classification model trained on human labeled data to get the ESCI labels.
- **PR@100:**: This metric computes the purchase recall of queries in top-100 retrieved results, i.e., the percentage of query purchases that the model is able to retrieve in top-100 results during testing period.

Table 1: Comparison of ESCI@100 and Purchase Recall@100 of SMOCC with baselines on internal dataset. Each number represents the change in percentage over Random Negatives baseline

Method	E	S	C	I	PR@100
SMOCC-EM	+12.37	+1.17	-5.46	-8.08	+9.76
SMOCC-QS	+9.28	+2.67	-4.87	-7.08	+8.32
DROCC [9]	+8.41	+2.15	-4.12	-6.91	+7.89
MoCo [4]	+7.41	+3.12	-2.91	-6.76	+6.02
Hard Negatives	+6.37	+2.27	-2.82	-5.82	+6.56
InfoNCE [3]	+5.89	+2.81	-2.52	-4.31	+5.12

4.3 Baselines

We perform exhaustive evaluation of proposed SMOCC approaches against following baselines -

Random Negatives: In this baseline, we randomly selects 3 negative products for a relevant product to the query, and train the model with euclidean distance based MSE loss. This is similar to pre-training step in SMOCC.

Hard Negatives: This bi-encoder model is similar to [23] but instead of a two part hinge loss, we use a hard negative triplet loss as we observed that it performs much better empirically. In this approach also, we warm-up the training of bi-encoder models with Euclidean distance based MSE loss. Next, we train the model with a ranking based triplet loss (defined in SMOCC). For every query q_i and its relevant product p_i in the batch, we compute hard negatives within batch on the fly and compute the loss. Hard negative for query q_i is the product $\tilde{p_i}$ which lie closest to it in the batch in terms of Euclidean distance and $i \neq j$.

InfoNCE [3]: In this methodology, all the products $(p_j|j \neq i)$ in a minibatch for a query q_i are considered as negatives and all the queries $(q_k|k \neq i)$ in a mini batch are treated as negatives for the product p_i . There is no sampling of negatives explicitly. If batch size is N, we treat the other $2(N-1)$ augmented examples within a minibatch as negative examples for given a positive pair. The loss function used here is the normalized temperature-scaled cross entropy loss with temperature $\tau = 0.2$.

MoCo [4]: MoCo computes hard negative based upon dictionary look-up and is mainly used in the visual representation learning. It builds a dynamic dictionary with a queue and a moving-averaged encoder. This facilitates MoCo to perform on-the-fly contrastive learning.

DROCC [9]: This method is explained in Section 3.1

4.4 Results and Discussion

Table 1 shows the main results of our experiments. We report all the numbers relative to a random negatives baseline, i.e. absolute percentage points improvement over random negatives baseline because of legal restrictions. We can see that the performance of the SMOCC methods are significantly better at retrieving the exact matches than the baselines. In particular SMOCC-EM gives up to 5 percentage points better exacts than the best baseline. In addition to that, SMOCC methods also reduce the irrelevant matches by 1.2 percentage points compared to the best baseline, which significantly improves the user experience.

Table 2: Comparison of ESCI@5 of SMOCC with baselines on public dataset. Each number represents the change in percentage over Random Negatives baseline

Method	E	S	C	I
SMOCC-EM	+9.31	+2.25	-2.41	-6.32
SMOCC-QS	+7.36	+3.12	-2.75	-5.28
DROCC	+6.81	+2.97	-3.01	-5.91
MoCo	+5.76	+1.81	-2.36	-4.86
Hard Negatives	+4.63	+1.59	-1.81	-4.79
InfoNCE	+4.51	+1.73	-3.56	-3.89

Table 3: Effect of Curriculum Learning (CL) on SMOCC compared to random shuffling of data. Each number represents the percentage change in ESCI and Purchase Recall over Random Negatives baseline

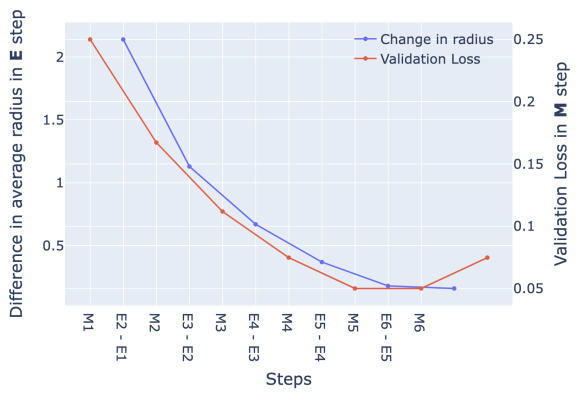
Method	Learning	E	S	C	I	PR@100
SMOCC -EM	CL	+12.37	+1.17	-5.46	-8.08	+9.76
	Random	+7.47	+1.17	-1.82	-6.82	+7.95
SMOCC -QS	CL	+9.28	+2.67	-4.87	-7.08	+8.32
	Random	+2.37	-0.66	+3.82	-5.43	+4.41

Table 2 shows the results on KDD Cup '22 dataset. As noted above, we choose to present ESCI@5 because of limited availability of exact products per query. We note a similar trend with the public dataset as with the internal data. SMOCC-EM outperforms the baselines significantly. SMOCC-EM achieves up to 3.5 percentage points better exacts than the best performing baseline, while reducing the irrelevants by 1.5 percentage points.

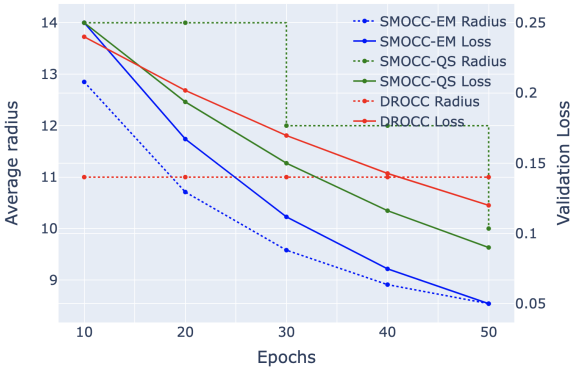
As noted in Section 3.7, we use curriculum learning to improve the performance of SMOCC methods. Table 3 shows the effect of curriculum learning on the performance of both the SMOCC methods. For curriculum learning, we perform the pre-training as before for 10 epochs. We then split the data into k bins based on ranges of query specificity for SMOCC-QS, or based on ranges of radii for SMOCC-EM. We present the results for $k = 3$. The data in each bin is trained for 3×10 epochs, which is equivalent to training on the entire data for 40 epochs. Using curriculum learning provides a boost of 7 percent more exacts for SMOCC-QS and 5 percent more exacts for SMOCC-EM. It is interesting to note that SMOCC-QS with curriculum learning outperforms SMOCC-EM without curriculum learning. This is because curriculum learning helps the model generalize better and reduces the impact of potentially noisy labels obtained from other models.

4.5 Effect of Radius and Convergence of SMOCC-EM

We now show the effect of varying the radius on performance of SMOCC methods. Figure 2a shows how the radius converges with each E step of SMOCC-EM. We notice that the difference in the



(a) Convergence of radius in E step and loss in corresponding M step SMOCC-EM.



(b) Radius used for SMOCC-EM, SMOCC-QS and DROCC, and their corresponding losses.

Figure 2: Effect of varying radius on SMOCC methods

radius between each **E** step decreases with iterations. This means that the distances between the query and the relevant product embeddings decrease with each each **M** step, and thus the regression model returns a smaller radius. We also note that after certain **E** steps, the radius shrinks so low that some relevant product embeddings lie outside of the radius. This leads to an increase in the validation loss. We use this as an indication to terminate the EM steps.

Figure 2b shows the effect of varying the radius on the performance of SMOCC methods. This shows the power of SMOCC algorithms over the fixed radius of DROCC. We also notice that using radius values provided by the regression model significantly improves the convergence rate as well as the value of the loss for SMOCC-EM when compared with stepped decrease of SMOCC-QS.

4.6 Effect of Quantization of SMOCC-QS

In this section we will show the effect of varying the quantization of query specificity in SMOCC-QS. We show the results for 3 bins, 5 bins and 7 bins. We notice that the performance increases when increasing the bins from 3 to 5, but the change from 5 to 7 is insignificant.

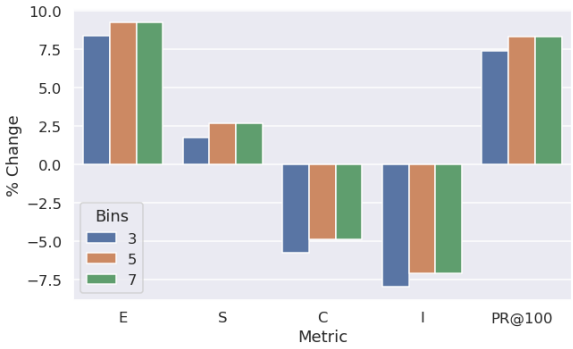


Figure 3: Effect of varying the number of bins on the performance of SMOCC-QS.

Table 4: ESCI comparison of sBERT based SMOCC

Method	E	S	C	I	PR@100
SMOCC-EM	+33.36	-11.67	-10.45	-11.14	+12.92
SMOCC-QS	+30.28	-10.11	-10.11	-10.14	+12.21
DROCC	+26.47	-9.32	-9.21	-10.03	+8.65

4.7 BERT based SMOCC

The goal of developing SMOCC was to create a fast semantic matching model that can be used in real-world applications requiring low latency. In this section, we present the results of our experiments on BERT-based SMOCC. These use the same algorithms presented in Section 3 but use BERT as the underlying model instead of a two-tower model presented. Specifically, we use sentence-BERT trained on MS MARCO [22]. MS MARCO is a large scale information retrieval corpus that was created based on real user search queries using Bing search engine. Table 4 shows the results of our experiments on BERT-based SMOCC. We notice that the performance of BERT-based SMOCC is significantly better than the performance of shallow non-BERT based models, but the latency is prohibitively large for any real time applications.

5 CONCLUSIONS

In this paper we presented a novel approach to semantic matching for product search based on one-class classification called SMOCC. SMOCC works in two steps: 1) It generates the representation of a negative product using adversarial search, and 2) It uses the negative product along with the query and positive product to train the model using triplet loss. We also propose two novel methods for determining the radius when performing adversarial search. This gives us two variants of our methods: SMOCC-QS uses quantized radius values based on query specificity and SMOCC-EM learns the optimal radius using a regression model and expectation maximization paradigm. We show that both of these methods outperform the state of the art baselines by increasing the exact matches by up to 5%, reducing the irrelevant matches by 1.8% and increasing the purchase recall at 100 by more than 3%. We also perform extensive analysis showing the impact of curriculum learning and radius selection, and how it results in improved performance.

REFERENCES

[1] Arindam Bhattacharya, Sumanth Varambally, Amitabha Bagchi, and Srikanta Bedathur. 2021. Fast One-Class Classification Using Class Boundary-Preserving Random Projections. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD '21)*. Association for Computing Machinery, New York, NY, USA, 66–74.

[2] Wei-Cheng Chang, Daniel Jiang, Hsiang-Fu Yu, Choon Hui Teo, Jiong Zhang, Kai Zhong, Kedarnath Kolluri, Qie Hu, Nikhil Shandilya, Vyacheslav Ievgrafov, Japinder Singh, and Inderjit S. Dhillon. 2021. Extreme Multi-Label Learning for Semantic Matching in Product Search.

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML'20*.

[4] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. 2020. Improved Baselines with Momentum Contrastive Learning. *arXiv preprint arXiv:2003.04297* (2020).

[5] 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://www.aclweb.org/anthology/N19-1423

[6] Miao Fan, Jiacheng Guo, Shuai Zhu, Shuo Miao, Mingming Sun, and Ping Li. 2019. MOBIUS: Towards the Next Generation of Query-Ad Matching in Baidu's Sponsored 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, 2509–2517.

[7] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings.

[8] Markus Goldstein and Andreas R. Dengel. 2012. Histogram-based Outlier Score (HBOS): A fast Unsupervised Anomaly Detection Algorithm.

[9] Sachin Goyal, Aditi Raghunathan, Moksh Jain, Harsha Vardhan Simhadri, and Prateek Jain. 2020. DROCC: Deep Robust One-Class Classification. In *Proceedings of the 37th International Conference on Machine Learning (PMLR '20)*. PMLR, Cambridge MA, USA, 3711–3721.

[10] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2019. Momentum Contrast for Unsupervised Visual Representation Learning. https://arxiv.org/abs/1911.05722

[11] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-Based Retrieval in Facebook Search. In *KDD '20*.

[12] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based Retrieval in Facebook Search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM.

[13] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management* (San Francisco, California, USA) *(CIKM '13)*. Association for Computing Machinery, New York, NY, USA, 2333–2338.

[14] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. 2020. Hard Negative Mixing for Contrastive Learning. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 21798–21809. https://proceedings.neurips.cc/paper/2020/file/f7cade80b7cc92b991cf4d2806d6bd78-Paper.pdf

[15] Chaozhuo Li, Bochen Pang, Yuming Liu, Hao Sun, Zheng Liu, Xing Xie, Tianqi Yang, Yanling Cui, Liangjie Zhang, and Qi Zhang. 2021. AdsGNN: Behavior-Graph Augmented Relevance Modeling in Sponsored Search. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[16] Yiding Liu, Weixue Lu, Suqi Cheng, Daiting Shi, Shuaiqiang Wang, Zhicong Cheng, and Dawei Yin. 2021. Pre-Trained Language Model for Web-Scale Retrieval in Baidu Search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 3365–3375.

[17] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv:1907.11692 [cs.CL]

[18] Hanqing Lu, Youna Hu, Tong Zhao, Tony Wu, Yiwei Song, and Bing Yin. 2021. Graph-based Multilingual Product Retrieval in E-Commerce Search. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Papers*.

[19] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards Deep Learning Models Resistant to Adversarial Attacks.

[20] Yu. A. Malkov and D. A. Yashunin. 2018. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. arXiv:1603.09320 [cs.DS]

[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs.CL]

[22] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. (2016). https://www.microsoft.com/en-us/research/publication/ms-marco-human-generated-machine-reading-comprehension-dataset/

[23] 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 and Data Mining*.

[24] Pramuditha Perera, Ramesh Nallapati, and Bing Xiang. 2019. OCGAN: One-Class Novelty Detection Using GANs With Constrained Latent Representations. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 2893–2901.

[25] Tomás Pevný. 2015. Loda: Lightweight on-line detector of anomalies. *Machine Learning* 102 (2015), 275–304.

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

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

[28] Lukas Ruff, Nico Görnitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Robert A. Vandermeulen, Alexander Binder, Emmanuel Müller, and M. Kloft. 2018. Deep One-Class Classification. In *ICML*.

[29] Lukas Ruff, Jacob R. Kauffmann, Robert A. Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G. Dietterich, and Klaus-Robert Muller. 2021. A Unifying Review of Deep and Shallow Anomaly Detection. *Proc. IEEE* 109 (2021), 756–795.

[30] Ankith M S, Sourab Mangrulkar, and Vivek Sembium. 2022. HISS: A novel hybrid inference architecture in embedding based product sourcing using knowledge distillation. In *SIGIR 2022*. https://www.amazon.science/publications/hiss-a-novel-hybrid-inference-architecture-in-embedding-based-product-sourcing-using-knowledge-distillation

[31] Bernhard Schölkopf, Robert C. Williamson, Alex Smola, John Shawe-Taylor, and John C. Platt. 1999. Support Vector Method for Novelty Detection. In *NIPS*.

[32] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

[33] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Gregoire Mesnil. 2014. A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. In *CIKM*. https://www.microsoft.com/en-us/research/publication/a-latent-semantic-model-with-convolutional-pooling-structure-for-information-retrieval/

[34] David M. J. Tax and Robert P. W. Duin. 2004. Support Vector Data Description. *Machine Learning* 54 (2004), 45–66.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need.

[36] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval.

[37] Jinyu Yang, Jiali Duan, Son Tran, Yi Xu, Sampath Chanda, Liqun Chen, Belinda Zeng, Trishul Chilimbi, and Junzhou Huang. 2022. Vision-Language Pre-Training With Triple Contrastive Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 15671–15680.

[38] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing Dense Retrieval Model Training with Hard Negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 1503–1512.

[39] Hongchun Zhang, Tianyi Wang, Xiaonan Meng, Yi Hu, and Hao Wang. 2019. Improving Semantic Matching via Multi-Task Learning in E-Commerce. In *Proceedings of the SIGIR 2019 Workshop on eCommerce (SIGIR 2019 eCom)*.

[40] Yanzhao Zhang, Richong Zhang, Samuel Mensah, Xudong Liu, and Yongyi Mao. 2022. Unsupervised Sentence Representation via Contrastive Learning with Mixing Negatives. *Proceedings of the AAAI Conference on Artificial Intelligence* 36, 10 (Jun. 2022), 11730–11738.