ANTHEM: Attentive Hyperbolic Entity Model for Product Search

Nurendra Choudhary[1], Nikhil Rao[2], Sumeet Katariya[2], Karthik Subbian[2], Chandan K. Reddy[1,2]
[1]Department of Computer Science, Virginia Tech, Arlington, VA, USA
[2]Amazon, Palo Alto, CA, USA
nurendra@vt.edu,{nikhilsr,katsumee,ksubbian}@amazon.com,reddy@cs.vt.edu

ABSTRACT

Product search is a fundamentally challenging problem due to the large-size of product catalogues and the complexity of extracting semantic information from products. In addition to this, the black-box nature of most search systems also hamper a smooth customer experience. Current approaches in this area utilize lexical and semantic product information to match user queries against products. However, these models lack (i) a hierarchical query representation, (ii) a mechanism to detect and capture inter-entity relationships within a query, and (iii) a query composition method specific to e-commerce domain. To address these challenges, in this paper, we propose an AtteNTive Hyperbolic Entity Model (ANTHEM), a novel attention-based product search framework that models query entities as two-vector hyperboloids, learns inter-entity intersections and utilizes attention to unionize individual entities and inter-entity intersections to predict product matches from the search space. ANTHEM utilizes the first and second vector of hyperboloids to determine the query's semantic position and to tune its surrounding search volume, respectively. The attention networks capture the significance of intra-entity and inter-entity intersections to the final query space. Additionally, we provide a mechanism to comprehend ANTHEM and understand the significance of query entities towards the final resultant products. We evaluate the performance of our model on real data collected from popular e-commerce sites. Our experimental study on the offline data demonstrates compelling evidence of ANTHEM's superior performance over state-of-the-art product search methods with an improvement of more than 10% on various metrics. We also demonstrate the quality of ANTHEM's query encoder using a query matching task.

CCS CONCEPTS

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

KEYWORDS

E-commerce, product search, query representation, hyperbolic space

ACM Reference Format:
Nurendra Choudhary[1], Nikhil Rao[2], Sumeet Katariya[2], Karthik Subbian[2], Chandan K. Reddy[1,2]. 2022. ANTHEM: Attentive Hyperbolic Entity Model for Product Search. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3488560.3498456

WSDM '22, February 21–25, 2022, Tempe, AZ, USA.

ACM ISBN 978-1-4503-9132-0/22/02.
https://doi.org/10.1145/3488560.3498456

1 INTRODUCTION

In e-commerce, a majority of customers begin their journey on websites via a product search functionality. When a user searches for an item, they would potentially see a ranked (or tiled) list of products that best match the intent of the query. User queries are often vague, broad, short, and do not follow any specific natural language structure. Additionally, the catalogue for e-commerce websites is ever growing, and rapidly changing. The above reasons compelled with the need to show an array of related, yet complementary and substitute items makes it hard to match and show appropriate results to these queries.

Product search heavily influences both user behavior and experience. Not only do unsuitable results upset user experience, but the black-box nature of current search systems does not allow researchers to gain insight into the problems of querying process. Thus, the only source of feedback is the final set of search results. Being able to interpret these search results will allow researchers to gain insight into the system and subsequently improve both their query comprehension and query processing methods [10, 24, 33].

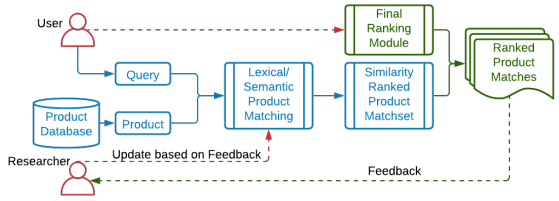


Figure 1: Product search framework. This paper focuses on improving the Product Matching module, optimized for recall in semantic matching and precision in ranking.

Current search frameworks, as shown in Figure 1, include two major modules for retrieving the product matches for a given input query [28]; (i) a matching phase that generates a set of items deemed appropriate to the query, and (ii) a ranking phase that ranks these items in a certain order of suitability. Traditional approaches for matching [21, 41] lexically match queries to an inverted index to retrieve all products that contain the query's words. Such methods do not understand the query's semantic intent of hypernyms (*sneakers* vs *running shoes*), synonyms (*blue* vs *sapphire*) and antonyms (*sugar-free* vs *sugary*). Additionally, these methods, generally include lemmatization as a preprocessing step, which loses morphological information (*running* vs *run*) and cannot capture out-of-vocabulary (OOV) words. Recent approaches [18, 28] learn a joint query-product matching model with character-trigram tokens (instead of lemmatized words) as inputs to deep learning encoders. The character trigrams allow morphological complexity and handle the OOV words [4] while the deep learning encoders capture

semantic information from both the query and products. However, these approaches are limited due to the following challenges:

(1) **Hierarchical structure**: Existing methods do not leverage the inherent hierarchy present in the product catalogues. This motivates the need for using hyperbolic spaces that better conform to the latent anatomy of product data compared to their Euclidean counterparts [11].
(2) **Dynamic query space**: Current product matching approaches utilize a fixed threshold (top-K retrieval) to return items in the match set. However, general queries like *men shoes* should match onto a larger portion of the catalogue than narrower queries like *nike men's red running shoes*. This necessitates the query representation to be spatially-aware, i.e., covering a broader space of items for general queries.
(3) **User query composition**: Inspired by text processing, current methods compose queries as a sequence of semantic tokens, e.g., $P(nike\ adidas) = P(adidas|nike)P(nike)$. However, the product queries are, generally, composed of independent tokens with hierarchical connections. Thus, query composition depends upon capturing the complex hierarchical intersection/union between item tokens and their individual semantic information, e.g., $P(nike\ adidas) = P(nike \cup adidas)P(nike)P(adidas)$.

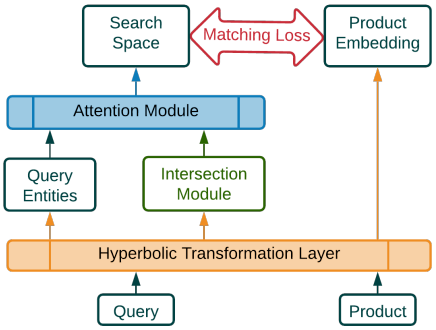


Figure 2: Overview of the proposed ANTHEM model. The query entities are encoded in the hyperbolic space as hyperboloids. Attention over the individual entities and their intersections results in the final search space which is matched against the product embeddings.

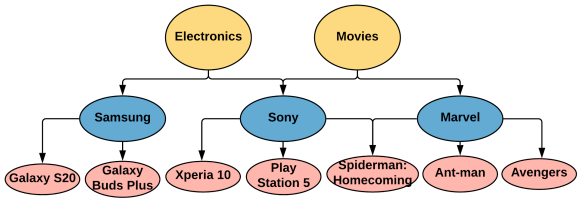


Figure 3: Hierarchy of products in the catalogue.

To alleviate the above challenges, we propose AtteNTive Hyperbolic Entity Model (ANTHEM), illustrated in Figure 2, a novel attentive joint-learning framework that learns spatially-aware query representations. The representations capture the query's hierarchical relations, and learns geometric operations (union and intersection) to match them with products. In addition to this, we provide an explainability mechanism that allows researchers to understand the ANTHEM's internals and provide explainable search results.

In most of the industrial e-commerce settings, one would observe that products invariably lie in a hierarchy (see Figure 3) and relations between them are either hierarchical ($nike\ shoes \subset shoes$) or independent ($nike \cap adidas = \varnothing$). Thus, unlike current approaches, we aim to preserve the hierarchical relations in addition to the semantic features. Hyperbolic spaces have proven to be more effective than Euclidean spaces at modeling such hierarchical relations [7, 11, 34]. Thus, we learn representations of our products as vectors in a hyperbolic space. Consequentially, we also need to learn our query representations in the hyperbolic space. However, the query's search space varies based on its broadness and relies on hierarchical relationships between its entities. To handle the query broadness, ANTHEM models the queries as hyperboloids with two hyperbolic vectors; the center and limit. The center defines the location of a query hyperboloid in the hyperbolic semantic space and its limit determines the search space (or volume) around the center. Thus, ANTHEM is capable of learning variable search volumes depending on the scope of a query. ANTHEM applies attention over individual tokens and their intersections to capture the significance of hierarchical relations, respectively, to the final search results in a hyperbolic Poincaré ball of unit radius. The activation units of the attention layers help analyze and explain our model's search results for a query, thus making ANTHEM more interpretable compared to other methods. To the best of our knowledge, there is no existing work that models spatially-aware queries or utilizes attention in hyperbolic spaces to capture hierarchical relations.

Through a variety of experiments, we show that ANTHEM outperforms state-of-the-art baselines in product search, while being interpretable. To understand the semantic capabilities of ANTHEM's query encoder in isolation, we test it on query-matching classification. In addition, we analyze the contribution of ANTHEM's individual components to the overall performance through an ablation study. To summarize, the contributions of this paper include:

(1) A novel product search framework, AtteNTive Hyperbolic Entity Model (ANTHEM) that utilizes token intersection/union and attention networks to compose queries as spatially-aware hyperboloids in a Poincaré ball, i.e., the query broadness is captured by the volume of hyperboloids.
(2) A mechanism that utilizes attention units' activation to understand the internal working of ANTHEM and explain its product search mechanism on sample queries.
(3) Analysis of ANTHEM's isolated query encoder and its ability to capture significant semantic features through the task of query matching on a popular e-commerce website.
(4) An extensive set of empirical evaluation to study the performance of ANTHEM as a product search engine on a real-world consumer behavior dataset retrieved from a popular e-commerce website against state-of-the-art baselines.

The rest of this paper is organized as follows: Section 2 discusses the relevant background. Section 3 formulates the product search problem and explains the proposed ANTHEM framework. In Section 4, we describe the real-world e-commerce datasets, state-of-the-art baselines and evaluation metrics used to evaluate the proposed model. Section 5 concludes the paper.

2 RELATED WORK

In this section, we review various product search and semantic matching methods studied in the literature. We will also discuss several existing techniques that are developed for hyperbolic spaces.

2.1 Product Search

Product search algorithms are primarily motivated by existing works on search engines in the fields of Information Retrieval (IR) and Natural Language Processing (NLP), where the goal is to learn semantic information from queries and documents. However, product search mainly differs from traditional search in two key aspects; (i) product titles tend to be shorter than documents and (ii) signals (i.e., purchase information) are sparser than click-through data. Traditional approaches [2, 41] rely on lexical information to construct inverted indices and match queries with product titles. However, these methods do not consider semantic information which is imperative to handle synonymy and hypernymy [32]. DESM model [26] successfully leverages Word2Vec [22] vectors to rank documents for web search. Furthermore, Diaz et al. [9] expand the queries with neighboring semantic words (synonyms). These methods are able to capture semantic information, but cannot handle out-of-vocabulary (OOV) words or typographical errors. Additionally, these bag-of-words model does not capture the sequential information of sentences. DSSM model [18] employs Siamese networks with shared weights to match query and documents with a contrastive loss function based on click-through data. Another significant addition to DSSM is the use of character-trigrams instead of complete words, which can effectively handle OOV words and typographical errors. C-DSSM [17] and R-DSSM [29] replace the dense layers in the DSSM model with convolutional and recurrent layers, respectively, in order to handle sequential information. Another line of work is DRMM [14] which utilizes a vocabulary interaction matrix in order to capture local semantic information. MatchPyramid [30] extends this approach further by using a convolution operation over the interaction matrix. DUET [23] combines the semantic and lexical matching benefits of DSSM and DRMM to obtain better results. However, these approaches show limited results on ad-hoc retrieval tasks. Nigam et al. [28] focus on product search on a practical e-commerce setting with a large number of query-product pairs and demonstrate the benefits of factorized models (DSSM) over interaction models (DRMM). Furthermore, Guo et al. [16] and Wang et al. [37] use grid-based search and meta-learning to improve the search experience. [27] use an adversarial model to learn hard-to-classify query-product pairs and [1, 20] improve product search by aggregating information from multiple languages.

2.2 Hyperbolic Spaces

One recent significant advancement in modeling hierarchical structures (such as the ones present in product catalogues) is the use of hyperbolic spaces. Ganea et al. [11] show that hyperbolic spaces better capture the inherent anatomy of hierarchical datasets compared to their Euclidean counterparts. The authors propose a Graph Neural Network and provide mathematical formulations for various gyrovector operations that are necessary for modeling network architectures. Gyrovector operations for Poincaré ball of radius c are Riemannian metric ($g_x^{\mathbb{H}}$), Möbius addition ($\oplus_c$), Möbius subtraction ($\ominus_c$), exponential map ($\exp_x^c$),and Möbius scalar product ($\odot_c$).

$$g_x^{\mathbb{H}} \coloneqq \lambda_x^2 \; g^{\mathbb{E}} \quad \text{where } \lambda_x \coloneqq \frac{2}{1-\|x\|^2} \tag{1}$$

$$x \oplus_c y \coloneqq \frac{\left(1+2c\langle x,y\rangle + c\|y\|^2\right)x + \left(1-c\|x\|^2\right)y}{1+2c\langle x,y\rangle + c^2\|x\|^2\|y\|^2} \tag{2}$$

$$x \ominus_c y \coloneqq x \oplus_c -y \tag{3}$$

$$\exp^c(v) \coloneqq \tanh\left(\sqrt{c}\frac{\lambda_0^c\|v\|}{2}\right)\frac{v}{\sqrt{c}\|v\|} \tag{4}$$

$$r \odot_c x \coloneqq \exp^c(r log_0^c(x)),\ \forall r \in \mathbb{R}, x \in \mathbb{H}_c^n \tag{5}$$

Here, $\coloneqq$ denotes assignment of Möbius operations. $g^{\mathbb{E}} = \mathbf{I}_n$ is the Euclidean identity metric tensor and $\|x\|$ is the Euclidean norm of x. λ_x is the conformal factor between the Euclidean and hyperbolic metric tensor. It is set to a conventional curvature of -1 [11].

Recently, hyperbolic spaces have been successfully leveraged to learn representations of graph networks [13] and knowledge graphs [5, 38]. Given the hierarchical nature of product catalogue, we extend hyperbolic functions to 2-dimensional geometries in ANTHEM to improve its performance for product search.

3 THE PROPOSED ANTHEM MODEL

In this section, we will first discuss the problem setup for product search and query-matching, and then describe the different layers and overall architecture of ANTHEM (illustrated in Figure 4).

3.1 Problem Setup and Notations

3.1.1 Product Search. Given the query set Q, for query $q \in Q$, let $\{s_1^+, s_2^+, .., s_{|S^+|}^+\} \in S^+$ and $\{s_1^-, s_2^-, .., s_{|S^-|}^-\} \in S^-$ be the set of products with positive and negative purchase signal, respectively, i.e., for a given query, positive samples were the items *purchased* and negative samples were *not purchased* by the user from the given search results. The primary goal of product search is to recommend S^+ for a query q. To achieve this, we train ANTHEM to optimize a model P_θ parameterized by θ such that:

$$\hat{y}_i^+ = P_\theta\left(s_i \in S^+ | q_i, \theta\right), \quad \hat{y}_i^- = P_\theta\left(s_i \in S^- | q_i, \theta\right), \quad \hat{y}_i^+ + \hat{y}_i^- = 1$$

$$\theta = \arg\min_\theta \left(-\sum_{i=1}^{|S^+|} y_i \log\left(\hat{y}_i^+\right) - \sum_{i=1}^{|S^-|} (1-y_i)\log\left(\hat{y}_i^-\right)\right)$$

where for a given query q_i, the probability of s_i being *purchased* and *not purchased* is provided by ANTHEM as $\hat{y}_i^+$ and $\hat{y}_i^-$, respectively. y_i denotes the Boolean ground truth purchase signal which is equal to 1, if product s_i is *purchased* for query q_i and 0, otherwise.

3.1.2 Query Matching. Due to the computational intensity of product search, most systems maintain pre-processed results for frequent queries [2]. Hence, it is more efficient to match a new query to an existing query result. In this problem, we aim to formulate query-matching as a multi-class classification task where the class labels define the similarity between query-pairs. Let $D = \{(q_i, q_j, y_{ij})\}$ be the training dataset, where q_i and q_j are queries and $y_{ij} \in Y$ is a multi-class categorical label denoting the relation between the queries; for example, query-pairs (q_a, q_b), (q_a, q_c) and (q_a, q_d) have labels $y_{ab} = class1$, $y_{ac} = class2$ and $y_{ad} = class3$.

The primary goal of query-matching is to predict class y_{ij} for a query-pair (q_i, q_j). To this end, ANTHEM applies the query encoder

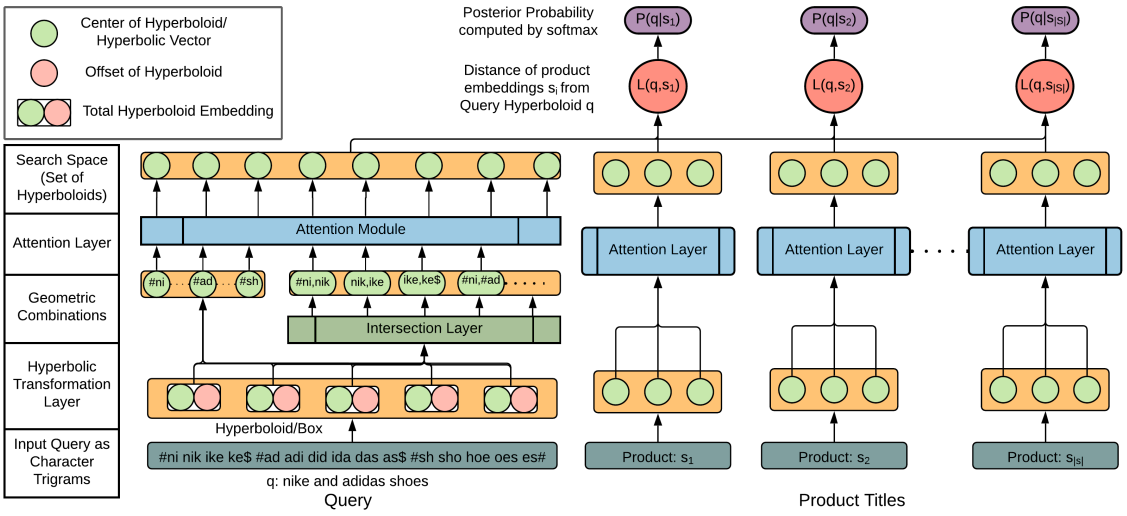


Figure 4: The overall architecture of our proposed ANTHEM model. The model encodes the queries' search space as a set of hyperboloids using attention over its entities and inter-entity intersections. The products are encoded as hyperbolic vectors with a self-attention on their char-trigrams. Finally, ANTHEM calculates the distance between product vectors and query hyperboloids and utilizes softmax to output a probability distribution over the products.

to optimize a model X_ϕ with parameters ϕ such that:

$$w_{y_c} = \frac{|(q_i, q_j, y_{ij}) \in D : y_{ij} = y_c|}{|D|}$$

$$\phi = \arg\min_{\phi} \left(- \sum_{i=1}^{|D|} \sum_{j=1}^{|Y|} w_{y_{ij}} y_{ij} \log \left(P_\phi \left(y_{ij} | q_i, q_j, \phi \right) \right) \right)$$

where w_{y_c} is the sampling weight of class y_c and $|.|$ denotes the number of elements in the set.

The queries $q \in Q$ and products $s \in S$ are natural language text encoded as m-hot sparse character trigrams ($q, s \in \mathbb{R}^{|V|}, |V| = 48,807$[1]). Applying them directly significantly increases the number of parameters, hence, we utilize an embedding layer ($Embedding : \mathbb{R}^{|V|} \rightarrow \mathbb{R}^d$, d is empirically set to 128) to compress the raw embeddings and represent the semantic position of character-trigram $q_i \in q$ as center ($cen(q_i) \in \mathbb{R}^d$). Additionally, we add a limit parameter ($lim(q_i) \in \mathbb{R}^d$) initialized at zero to capture the trigram's spatial awareness. Hence, the final Euclidean box embedding B_{q_i} is characterized by $(cen(q_i), lim(q_i)) \in \mathbb{R}^{2d}$:

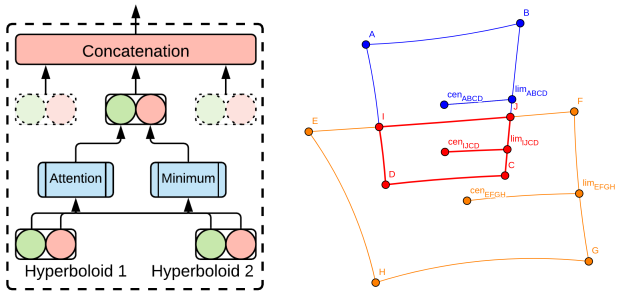
$$B_{q_i} \equiv \left\{ v \in \mathbb{R}^d : cen(q_i) - lim(q_i) \leq v \leq cen(q_i) + lim(q_i) \right\}$$

3.2 Layers of the ANTHEM

3.2.1 Hyperbolic Transformation Layer. The embedding layers learn representations in the Euclidean spaces. In order to model the hierarchical relationships between products more effectively, we transform the embeddings to a hyperbolic space. The transformation from Euclidean $(\mathbb{E}^n, g^{\mathbb{E}})$ to hyperbolic Poincaré ball $(\mathbb{H}^n, g^{\mathbb{H}})$ is an $O(1)$ operation defined by the Riemannian manifold $\mathbb{H}^n = \{x \in \mathbb{R}^n : \|x\| < 1\}$ and metric, $g^{\mathbb{H}}$ as given in Eq. (1). The Euclidean box B_{q_i} is transformed to a hyperboloid $H_{q_i} \in \mathbb{H}^{2d}$ with function $f_{hyp}(B_{q_i}) = H_{q_i}$ as:

$$H_{q_i} \equiv \left\{ v \in \mathbb{H}^d : g^{\mathbb{H}}_{cen(q_i)} \ominus_c g^{\mathbb{H}}_{lim(q_i)} \leq v \leq g^{\mathbb{H}}_{cen(q_i)} \oplus_c g^{\mathbb{H}}_{lim(q_i)} \right\}$$

where the final hyperboloid H_{q_i}, defined by a 2-vector enclosure, is a hyperbolic counterpart of a Euclidean rectangle.



(a) Intersection Layer **(b) Intersection of Hyperboloids**

Figure 5: Intersection Layer in ANTHEM and its interpretation in the hyperbolic space. (a) Intersection Layer in ANTHEM. Centers are aggregated using Attention and limits are aggregated with a Minimum layer. (b) Hyperboloid IJCD is the intersection of Hyperboloids ABCD and EFGH.

3.2.2 Intersection Layer. The customer intent in a query may not always be the union of its entities and could also mean an intersection between some parts of it, e.g., the query *nike shoes* is an intersection of brand entity *nike* and object entity *shoes*. To solve this, we learn the relation between different entities through intersection operation on 2d-spaces, previously defined for Euclidean space in Query2Box [31]. We extend the given Euclidean intersection function to the hyperbolic space and define the intersection of

[1] 48,807 is the total number of character trigrams with English characters and numbers.

queries $q_{ij} = \{q_i, q_j\}$ as $H_{q_{ij}}$:

$$H_{q_{ij}} = \left(cen\left(H_{q_{ij}}\right), lim\left(H_{q_{ij}}\right)\right) \tag{6}$$

$$cen\left(H_{q_{ij}}\right) = \sum_{n=i,j} a_n \odot_c cen\left(H_{q_n}\right); \quad a_n = \frac{exp^c(f(H_{q_n}))}{\sum_n exp^c(f(H_{q_n}))} \tag{7}$$

$$lim\left(H_{q_{ij}}\right) = Min\left(lim\left(H_{q_i}\right), lim\left(H_{q_j}\right)\right) \tag{8}$$

where $\odot_c$ is the Möbius scalar product, $f(.) : \mathbb{H}^{2d} \rightarrow \mathbb{H}^d$ is the multilayer perceptron (MLP), $\sigma(.)$ is the sigmoid function, $Min(.)$ and $exp_0^c(.)$ are the element-wise minimum and Möbius exponentiation functions, respectively. The new intersection center and limit (shown in Figure 5) are aggregated by an attention layer [35] over the centers and shrinking the limits with a minimum over queries. The main intuition here is that the semantic position (or) center (cen) of a set's intersection is the weighted sum of its entities and the space boundary (or) limit (lim) is the least of all the entities' boundaries.

3.3 Query Search Model

For a query containing n character trigrams, the individual query entity embeddings $H_q = \{H_1, H_2, ..., H_n\} \in \mathbb{H}^{2d}$ and intersection embeddings $H_{q_\cap} = \{H_{11}, H_{12}, ..., H_{nn}\} \in \mathbb{H}^{2d}$ do not contribute equally to the final search hyperboloid. We capture this varying significance to scale the embeddings using self-attention networks [35]. The hyperboloid entity $H_i \in H_q \cup H_{q_\cap}$ is scaled to $e_i \in \mathbb{H}^{2d}$ with function f_{att}:

$$f_{att}\left(H_i\right) = e_i = \sum_j \frac{exp^c(\alpha_{ij})}{\sum_i exp^c(\alpha_{ij})} H_i; \quad \alpha_{ij} = \frac{H_i^T H_j}{\sqrt{4d}} \tag{9}$$

Given a query q containing m character-trigram entities, the query encoder returns a search space $QS(q)$ which is a set of m and m^2 attention-scaled single and intersection hyperboloids, i.e., $QS(q) = e_1, .., e_i, .., e_{m(m+1)}$. Product s is encoded with a traditional attention network in the hyperbolic space [13] (shown in Figure 4). For a product s, the product encoder returns a hyperbolic vector $s \in \mathbb{H}^d$. Thus, unlike traditional approaches [18, 28], our query encoding is a set of hyperboloids and the product encoding is a hyperbolic vector. Thus, we need a specialized loss function that checks if the product vector is inside the query hyperboloids. The loss function L needs to capture the distance between s and $QS(q)$, which is simply the distance between s and the nearest hyperboloid in QS. Hence, we design our distance function as:

$$dist(s, QS(q)) = Min\left(\left\{d_{hyp}\left(H_i, s\right)\right\} \forall H_i \in QS(q)\right) \tag{10}$$

$$d_{hyp}\left(s, H_i\right) = d_{out}\left(s, H_i\right) \oplus_c \gamma d_{in}\left(s, H_i\right) \tag{11}$$

$$d_{out}\left(s, H_i\right) = \|Max(d_{\mathbb{H}}(s, H_{i_{max}}), 0) + Max(d_{\mathbb{H}}(H_{i_{min}}, s), 0)\|_1$$

$$d_{in}((s, H_i) = \|cen(H_i) \ominus_c Min(H_{i_{max}}, Max(H_{i_{min}}, s))\|_1$$

$$H_{i_{min}} = cen(H_i) \ominus_c lim(H_i), \quad H_{i_{max}} = cen(H_i) \oplus_c lim(H_i)$$

$$d_{\mathbb{H}}(x, y) = \cosh^{-1}\left(1 + 2\frac{\|x - y\|^2}{\left(1 - \|x\|^2\right)\left(1 - \|y\|^2\right)}\right)$$

where $d_{\mathbb{H}}(x, y)$ is the hyperbolic distance between hyperbolic vectors x and y. d_{in} and d_{out} is the distance of vector from the center and limits of the hyperboloid, respectively. γ is a scalar weight given to d_{in}. $\gamma = 0$ implies a hard hyperboloid limit border and all vectors are either considered inside or outside, whereas, $\gamma = 1$ implies no hyperboloid limit, thus, d_{hyp} is the distance between hyperboloid's center and a product vector. For our experiments, we set γ to 0.5. Choudhary et al. [7] show that the super-linear nature of d_{hyp} increases density of entity clusters as we move down the hierarchy of a dataset, thus, increasing distance between different entities at the same level and decreasing distance between entities with the same parent. To convert the distance function's output to a probabilistic distribution, we use a softmax layer [12]. Additionally, the loss function needs to minimize the distance between QS and $s \in S^+$ and maximize it between QS and $s \in S^-$. Thus, the final loss for products ($S = S^+ \cup S^-$) is a cross-entropy function calculated as follows:

$$\hat{y_i} = P(s_i|q) = \frac{exp^c(dist(s_i, QS))}{\sum_{s_i \in S} exp^c(dist(s_i, QS))} \tag{12}$$

$$L(\hat{y}, y) = \left(-\sum_{i=1}^{|S|} y_i \log\left(\hat{y}_i\right) - (1 - y_i) \log\left(1 - \hat{y}_i\right)\right) \tag{13}$$

Algorithm 1: ANTHEM training for Product Search

Data: Training data $D = (q \in Q, s \in S, y \in \{0, 1\})$;
Output: Predictor P_θ;
Initialize model parameters θ;
while *not converged* **do**
  $l = 0$; # Initialize loss
  for $\{(q, s, y) \in D\}$ **do**
    $q \leftarrow Embedding_\theta(q)$, $s \leftarrow Embedding_\theta(s)$;
    # Encode query q
    $H_q = f_{hyp}(q)$;
    $H_{q_\cap} = \{H_i \cap H_j\} \forall i, j : 1 \rightarrow n$; using Eq. (6)
    $QS = f_{att}(H_q \cup H_{q_\cap})$ using Eq. (9)
    # Encode product s
    $H_s = f_{hyp}(s)$;
    $e_s = f_{att}(s)$;
    # Calculate distance and update Loss l
    $\hat{y} = \frac{exp^c(dist(s,QS))}{\sum_{s \in S} exp^c(dist(s,QS))}$; using Eqs. (10),(12)
    $l = l + L(\hat{y}, y)$; using Eq. (13)
    # Update θ with back-propagation
    $\theta \leftarrow \theta - \nabla_\theta l$;
  end
end
return P_θ

3.4 Implementation Details

We implemented ANTHEM using Keras [6] on eight Nvidia V100 GPUs. For gradient descent in hyperbolic space, we adopt Riemannian Adam [3] with an initial learning rate of 0.0001 and standard β values of 0.9 and 0.999 and apply ReLU activation function [25]. The sensitivity of ANTHEM to hyper-parameters is presented in

Appendix D. For our empirical studies, we learn hyperboloid embeddings of (2×128) dimensions (d=128). The maximum sequence length of character trigram entities from queries and products is set to 28 and 128, respectively. This is set to fit the model's parameters in our GPUs. The sequence limit of 28 completely covers $\approx 94\%$ of the queries in our datasets. Algorithm 1 provides the pseudo-code for training ANTHEM[2] on the task of product search[3].

4 EXPERIMENTAL SETUP

This section will provide various experimental studies that analyze ANTHEM's performance and compare with existing state-of-the-art baseline methods. We aim to answer the following research questions (RQs) in this paper.

- **RQ1:** Are the embeddings from proposed ANTHEM model better compared to those from state-of-the-art baseline methods?
- **RQ2:** Does ANTHEM's query encoder capture semantic and hierarchical features for query-matching task?
- **RQ3:** What is the contribution of individual components to the overall performance of ANTHEM?
- **RQ4:** Can we explain the search results produced by ANTHEM?

Table 1: Basic statistics of the datasets used in experiments.

Dataset	Class	Train	Valid	Test
E-commerce Product Search	# Positive	742,500	106,071	212,144
	# Negative	3,861,002	551,571	1,103,145
E-commerce Query Matching	# Exact	456,652	65,236	130,473
	# Substitute	41,372	5,962	12,284
	# Complement	4,483	651	1,271
	# Irrevelant	22,865	3,266	6,714
Public E-commerce Search Relevance	# Positive	4,786	684	1,367
	# Negative	9,582	1,369	2,738

4.1 Datasets

We conducted experiments on various product search and query-matching datasets. Table 1 presents additional details about the data distribution for these datasets.

- *E-commerce Product Search*[4]: This dataset consists of 6.6M query-product pairs (retrieved from a popular e-commerce website) with a purchase signal which is a Boolean indicator that denotes whether a product was purchased[5] or never purchased for a given query. The dataset is completely anonymized, and subsampled to enable efficient model training.
- *Public E-commerce Search Relevance*[6]: This publicly available dataset consists of 20K labeled query-product pairs (columns *query*, *product_title*) with a purchase signal collected from the following five e-commerce websites: *eBay*, *OverStock*, *Shop.com*, *Target*, and *Walmart*. The purchase signal is a Boolean indicator based on the column *relevance* that denotes if a product is relevant if $relevance > 0.5$, else the product is considered irrelevant.
- *E-commerce Query Matching*[4]: This dataset consists of 750K query-query pairs (q_i, q_j) with a matching class, retrieved from a popular e-commerce website. The matching class will be one of the following four classes: (i) *Exact*: q_i and q_j are exact matches and produce the same results, e.g., *ps4 games* and *playstation 4 games*. (ii) *Substitute:* Parts of q_i and q_j can be substituted with one another, e.g., *nike shoes* and *adidas shoes*. (iii) *Complement:* q_i and q_j are complementary in meaning, e.g., *phones* and *phone screen protectors*. (iv) *Unrelated:* q_i and q_j are completely unrelated to each other, e.g., *phones* and *shoes*.

[2]Our code: https://github.com/amazon-research/hyperbolic-embeddings
[3]Algorithm for query matching task is provided in Appendix C.
[4]Proprietary dataset
[5]To avoid anomalies, we only consider products purchased more than a predefined number of times.
[6]https://data.world/crowdflower/ecommerce-search-relevance

4.2 Baselines

To compare ANTHEM against the state-of-the-art frameworks, we select the following baselines based on previous research.

- **ARC-II** [17] utilizes a joint learning Siamese convolutional network to semantically match natural language sentences.
- **KNRM** [39] is another neural ranking model that uses a kernel pooling over cosine similarity between the query and document, followed by a dense layer to compute probabilistic scores.
- **DRMM** [14] is a neural ranking model that uses a histogram-like interaction vector to bin cosine similarity between the query and document into predefined intervals, followed by a dense layer to compute probabilistic scores.
- **aNMM** [40], similar to DRMM, computes a fixed-dimensional interaction vector by binning the cosine similarity between each of the query and document words. However, this model uses the total sum of the similarity between those word pairs as the features instead of using the counts of word-pairs.
- **MatchPyramid** [30] computes an interaction matrix between queries and document, and then passes it through CNN layers with dynamic pooling to compute sentence similarity.
- **C-DSSM** [32] is a twin-tower architecture that utilizes convolution networks to capture sequential information from character trigrams as inputs. This is currently the most scalable framework and applied in most of the product matching systems [28].
- **DUET** [23] combines the semantic and lexical matching strengths of C-DSSM and DRMM, in a deep convolutional architecture, to compute sentence similarity.
- **MV-LSTM** [36] employs multiple positional sentence representations to match sentences. The architecture aggregates an interaction matrix between different Bi-LSTM encoded positional sentence representations through multi-layer perceptrons.
- **BERT** [8] utilizes transformers to capture the co-dependence of different sentence units as attention weights. For this, BERT trains a language model by masking certain inputs. We adopt the large pre-trained BERT model and fine-tune it for our experiments.

The baselines are implemented in the Matchzoo framework [15] and the hyper-parameters are tuned using grid-search.

4.3 RQ1: Performance on Product Search

To analyze the efficacy of the query representations obtained from ANTHEM's query encoder, we compare it against the state-of-the-art baselines on different product search datasets. ANTHEM takes a query-product pair as input and outputs the probability that the product belongs to the query's search space ($P(s|q)$). The probability is calculated $\forall s \in S$ and the results are ranked to get the final search results. We evaluate our model using 5-fold cross validation on the following standard ranking metrics: Normalized Discounted Cummulative Gain (**NDCG@K**), Mean Average Precision (**MAP**),

Table 2: Performance comparison of the proposed ANTHEM architecture with several state-of-the-art baselines across proprietary and public product search datasets and evaluation metrics. E-ANTHEM is the Euclidean variant of ANTHEM without the Hyperbolic transformation layer. The results presented for the proprietary datasets are relative (in %) to the baseline ARC-II model. Exact evaluation results are presented for the public datasets. The best and second best results are highlighted in bold and underline, respectively. The symbol * indicates statistically significant improvement over BERT with a p-value ≤ 0.05.

Datasets	E-commerce Product Search (in %)					Public E-commerce Search Relevance (in %)				
Models	**NDCG@3**	**NDCG@5**	**NDCG@10**	**MAP**	**MRR**	**NDCG@3**	**NDCG@5**	**NDCG@10**	**MAP**	**MRR**
ARC-II	0	0	0	0	0	59.2	58.1	54.4	58.2	48.5
KNRM	12.5	12.8	15.0	12.8	16.7	66.6	65.5	62.6	65.6	56.6
DUET	13.1	13.3	15.4	13.4	14.7	66.9	65.8	62.8	66.0	55.6
DRMM	20.5	22.8	24.4	20.3	21.7	71.3	71.3	67.7	70.0	59.0
aNMM	21.0	23.9	26.8	20.3	22.9	71.6	72.0	69.0	70.0	59.6
MatchPyramid	25.6	25.6	26.8	25.0	34.7	74.3	72.9	69.0	72.8	65.3
C-DSSM	31.3	29.4	44.7	32.6	27.8	77.7	75.2	78.7	77.1	61.9
MV-LSTM	34.7	33.3	55.7	34.3	37.7	79.7	77.5	84.7	78.2	66.8
BERT	38.6	37.2	65.9	40.1	51.0	82.1	79.7	90.2	81.5	73.2
E-ANTHEM	49.4	46.7	66.7	51.2	62.9	88.5	85.2	90.7	88.0	79.0
ANTHEM	**51.1***	**48.9***	**80.9***	**53.5***	**65.4***	**89.5***	**86.5***	**98.4***	**89.3***	**80.2***

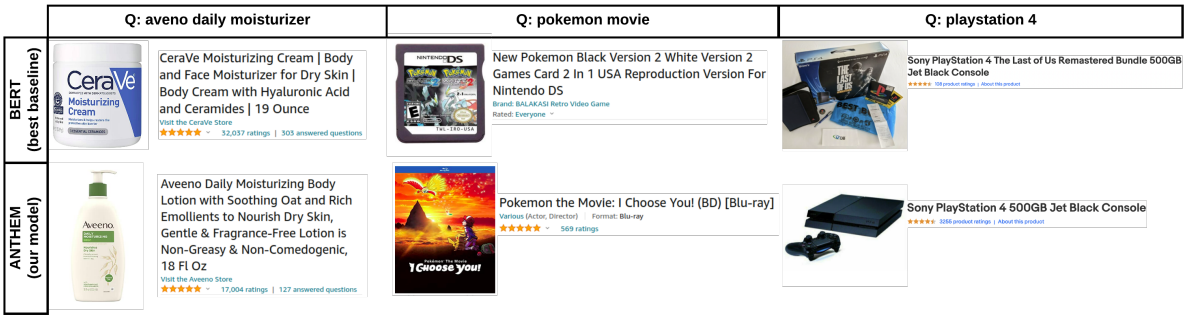


Figure 6: Example results for sample queries shown by our model and the best performing baseline. The results given are the third result for the query. The first two results are not shown here because they were equally appropriate for the query and the figure aims to show the differences between BERT and our model.

and Mean Reciprocal Rank (**MRR**). The datasets are split into training, valid and test sets of ratio 7:1:2, as given in Table 1. The results on the test set are presented in Table 2.

From the results, we observe that ANTHEM outperforms the state-of-the-art baselines across datasets by 10% – 15% in all the evaluation metrics. Additionally, we can notice that utilizing Euclidean spaces also improves the performance by ≈ 9%. This is empirical evidence that ANTHEM's spatially-aware query hyperboloids form better search space for E-commerce queries. Another point of note is that BERT (the best-performing baseline) has over 100M parameters, whereas, ANTHEM is able to achieve better results with only 6.37M parameters. In addition, we qualitatively analyze the results of our model and the state-of-the-art baselines. From the sample, shown in Figure 6, we observe that BERT is not able to capture the correct significance of brand intent ("aveno" in query and "aveeno" in products) from the query and emphasizes on the actual product type. ANTHEM, on the other hand, is able to provide more appropriate results due to the use of character-trigrams and inter-entity relations that are able to infer the importance of brand information and relation between "aveeno" and "moisturizer", respectively. In the second case, for the query *pokemon movie*, BERT focuses on the brand information and cannot differentiate the product type from the given title, thus, recommending an unsuitable product, whereas, the inter-entity relation in ANTHEM define the query as an intersection between "pokemon" and "movie" entities, consequentially, providing more appropriate results. In the last case, "playstation 4" is a short query with an ambiguous user intent. However, we notice that our model leverages hierarchical brand information to select the more pertinent product, whereas, BERT returns an improper result due to the ambiguity. This demonstrates the importance of hierarchical information in product search.

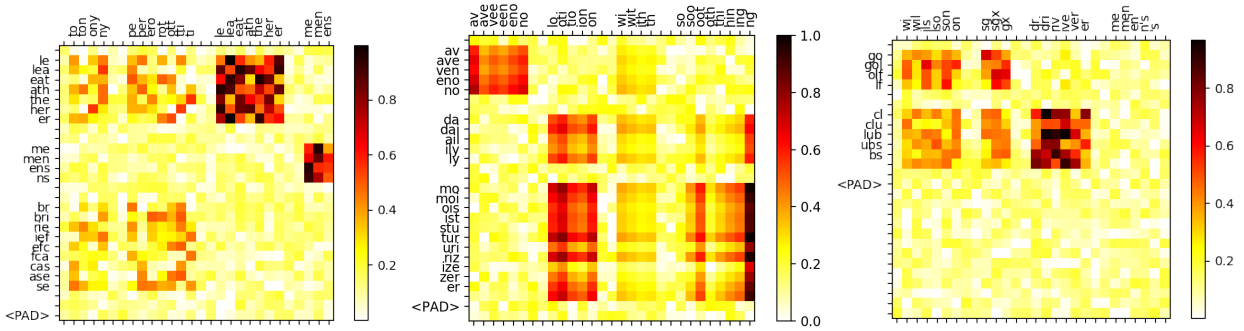
4.4 RQ2: Performance on Query Matching

To analyze the efficiency of our model's query encoder in isolation, we compare it against the state-of-the-art baselines on the query matching dataset. In this experiment, we pair the query encoder part of ANTHEM with a matching function (cosine similarity) in a siamese learning framework. The architecture takes a query pair as input and outputs the similarity between the queries. We evaluate our model with 5-fold cross validation on the standard classification metrics; **Accuracy**, **F-score**, and Area under ROC (**AUC**). The datasets are split into training, valid and test sets of ratio 7:1:2, as given in Table 1. The results are presented in Table 4.

From the results, we observe that ANTHEM is able to outperform the state-of-the-art baselines across datasets by 4% – 8% in the evaluation metrics. E-ANTHEM, utilizing Euclidean spaces also improves the performance by 3% – 6%. This is empirical evidence that ANTHEM's query encoder is able to capture the most significant semantic features.

4.5 RQ3: Ablation Study

In this experiment, we study the importance of different components that contribute to the overall performance of our proposed model. The components studied in our ablation experiments are:



(a) Mapping brands to products (b) Translating semantically similar phrases (c) Semantic/lexical matching

Figure 7: Explainability Study. Significance of query entities (y-axis) to the product entities (x-axis) analyzed through the final attention layer of the ANTHEM product search model. (a) ANTHEM is able to learn a matching from brand `tony perroti` to item tokens `leather` and `briefcase`, which enables better query-product matching. (b) ANTHEM is able to semantically map query term `daily moisturizer` to a lexically different term in the product `lotion`. (c) ANTHEM is able to leverage hierarchical brand from products `wilson sgx` and match to queries with no direct semantic/lexical similarity `golf clubs`.

Table 3: Ablation study. Performance comparison of the contributions from different components: Hyperbolic layer (H), Intersection layer (I), and Limit parameter (L). The results presented for the proprietary dataset are relative to the performance of first row (w/o L w/o I w/o H). 'w/o' stands for without. For public datasets, exact evaluation metrics are presented.

Datasets	E-commerce Product Search (in %)					Public E-commerce Search Relevance (in %)					E-commerce Query Matching (in %)		
Models	NDCG@3	NDCG@5	NDCG@10	MAP	MRR	NDCG@3	NDCG@5	NDCG@10	MAP	MRR	Accuracy	F-score	AUC
w/o L w/o I w/o H	0.0	0.0	0.0	0.0	0.0	41.0	40.9	28.5	39.2	31.2	0.0	0.0	0.0
w/o L w/o I	6.4	6.2	7.5	6.0	7.7	58.0	57.1	47.7	56.3	47.6	4.0	3.7	3.0
w/o I	23.4	22.9	36.8	24.5	30.2	77.2	75.0	74.2	76.2	67.4	54.8	54.8	55.2
ANTHEM	41.5	39.6	67.3	43.5	52.6	89.5	86.5	98.4	89.3	80.2	104.8	99.3	99.6

Table 4: Performance comparison of the proposed ANTHEM model with several state-of-the-art baselines on the E-commerce query matching dataset and evaluation metrics. The results presented are relative to the baseline ARC-II.

Models	Accuracy (in %)	F-score (in %)	AUC (in %)
ARC-II	0.0	0.0	0.0
KNRM	-4.1	-24.9	-19.1
DRMM	25.1	15.4	33.1
aNMM	-1.3	-8.6	4.0
MatchPyramid	-14.5	-17.8	-9.7
C-DSSM	21.2	21.7	30.1
DUET	-2.3	-4.7	0.9
MV-LSTM	71.1	21.2	48.9
BERT	40.1	33.5	54.7
E-ANTHEM	43.2	40.3	61.4
ANTHEM	**43.9**	**40.8**	**62.6**

(i) Hyperbolic layer (hierarchical features), (ii) Intersection layer (capturing inter-entity relations) and (iii) Limit parameter (spatial-awareness). The results of our experiments are presented in Table 3.

The results show that the Intersection layer and Limit parameter contribute $\approx 25\%$ to the overall performance of ANTHEM. Thus, we conclude that capturing inter-entity relationships in spatially-aware representations aid the performance of product search. Furthermore, removing the Hyperbolic layer decreases the performance by an additional $3\% - 8\%$ which shows the contribution of hierarchical information to the overall performance.

4.6 RQ4: Explainability Study

In this section, we analyze the internal working and significance of the query to the final results by utilizing the activation units of ANTHEM's attention layers. The attention units of a few sample queries are depicted in Figure 7 which provide a mechanism for researchers to understand the internal functions of our model. ANTHEM is able to match brands to products (Fig. 7a), translate semantically similar phrases (Fig. 7b) and leverage hierarchical information for semantic/lexical query-product matching (Fig. 7c). Thus, we conclude that concurrently utilizing product's hierarchical information (as Hyperboloids) and inter-product relation (as intersection) leads to better product representations. Also, such insights allow other researchers to independently analyze our model's suitability in their own applications and facilitate its integration.

5 CONCLUSION

In this paper, we presented ANTHEM, a novel product search framework that utilizes inter-token intersection/union and attention networks to encode query search spaces as spatially-aware hyperboloids in a Poincaré ball. We emphasized the utility of leveraging hierarchical information in product search and the need for spatially-aware query representations in the e-commerce domain. We performed an extensive set of empirical evaluation to study the performance and interpretability of our model as a product search engine on real-world query data collected from a popular e-commerce website. Finally, we validated the capability of our isolated query encoder in a query-matching task and analyzed the contribution of its components through an ablation study. Additionally, given the multitude of industrial applications, we also provide an explainability mechanism for researchers to analyze and integrate ANTHEM in their own architectures. We hope to provide a new perspective towards composing e-commerce queries as intersection and union of their individual tokens, instead of processing them as a sequence of tokens.

REFERENCES

[1] Aman Ahuja, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K Reddy. 2020. Language-Agnostic Representation Learning for Product Search on E-Commerce Platforms. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 7–15.

[2] A. Babenko and V. Lempitsky. 2015. The Inverted Multi-Index. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37, 6 (2015), 1247–1260.

[3] Gary Becigneul and Octavian-Eugen Ganea. 2019. Riemannian Adaptive Optimization Methods. In *International Conference on Learning Representations*.

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017).

[5] Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-Dimensional Hyperbolic Knowledge Graph Embeddings. In *58th Annual Meeting of the Association for Computational Linguistics*.

[6] Francois Chollet et al. 2015. *Keras*. https://github.com/fchollet/keras

[7] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K Reddy. 2021. Self-Supervised Hyperboloid Representations from Logical Queries over Knowledge Graphs. In *Proceedings of The Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

[9] Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query Expansion with Locally-Trained Word Embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

[10] Been Doshi-Velez, Finale; Kim. 2017. Towards A Rigorous Science of Interpretable Machine Learning. In *eprint arXiv:1702.08608*.

[11] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. 2018. Hyperbolic neural networks. In *Advances in neural information processing systems*.

[12] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*. Vol. 1. MIT press Cambridge.

[13] Caglar Gulcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, and Nando de Freitas. 2019. Hyperbolic Attention Networks. In *International Conference on Learning Representations*.

[14] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-Hoc Retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (Indianapolis, USA) *(CIKM '16)*. Association for Computing Machinery, New York, NY, USA.

[15] Jiafeng Guo, Yixing Fan, Xiang Ji, and Xueqi Cheng. 2019. MatchZoo: A Learning, Practicing, and Developing System for Neural Text Matching. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19)*. ACM, New York, NY, USA.

[16] Ruocheng Guo, Xiaoting Zhao, Adam Henderson, Liangjie Hong, and Huan Liu. 2020. Debiasing Grid-Based Product Search in E-Commerce. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA.

[17] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (Eds.), Vol. 27.

[18] 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. ACM International Conference on Information and Knowledge Management (CIKM).

[19] Byung-Do Kim and Peter E Rossi. 1994. Purchase frequency, sample selection, and price sensitivity: The heavy-user bias. *Marketing Letters* 5, 1 (1994), 57–67.

[20] Zheng Li, Mukul Kumar, William Headden, Bing Yin, Ying Wei, Yu Zhang, and Qiang Yang. 2020. Learn to Cross-lingual Transfer with Meta Graph Learning across Heterogeneous Languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2290–2301.

[21] Christopher D Manning, Hinrich Schütze, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Cambridge university press.

[22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*.

[23] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*.

[24] Christoph Molnar. 2018. A guide for making black box models explainable. *URL: https://christophm. github. io/interpretable-ml-book* (2018).

[25] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*.

[26] Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. 2016. Improving Document Ranking with Dual Word Embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web* (Montréal, Québec, Canada) *(WWW '16 Companion)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE.

[27] Thanh Nguyen, Nikhil Rao, and Karthik Subbian. 2020. Learning Robust Models for e-Commerce Product Search. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 6861–6869.

[28] Priyanka Nigam, Yiwei Song, Vijai Mohan, Vihan Lakshman, Weitian Ding, Ankit Shingavi, Choon Hui Teo, Hao Gu, and Bing Yin. 2019. Semantic product search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[29] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (2016).

[30] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text Matching as Image Recognition. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (Phoenix, Arizona) *(AAAI'16)*. AAAI Press.

[31] Hongyu Ren*, Weihua Hu*, and Jure Leskovec. 2020. Query2box: Reasoning over Knowledge Graphs in Vector Space Using Box Embeddings. In *International Conference on Learning Representations*.

[32] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd international conference on world wide web*.

[33] Ning Su, Jiyin He, Yiqun Liu, Min Zhang, and Shaoping Ma. 2018. User Intent, Behaviour, and Perceived Satisfaction in Product Search *(WSDM '18)*. Association for Computing Machinery, New York, NY, USA, 547–555. https://doi.org/10.1145/3159652.3159714

[34] Zequn Sun, Muhao Chen, Wei Hu, Chengming Wang, Jian Dai, and Wei Zhang. 2020. Knowledge Association with Hyperbolic Knowledge Graph Embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* (2017).

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

[37] Q. Wang, X. Liu, W. Liu, A. Liu, W. Liu, and T. Mei. 2020. MetaSearch: Incremental Product Search via Deep Meta-Learning. *IEEE Transactions on Image Processing* 29 (2020), 7549–7564.

[38] Shen Wang, Xiaokai Wei, Cicero dos Santos, Zhiguo Wang, Ramesh Nallapati, Andrew Arnold, Bing Xiang, and S Yu Philip. 2020. H2KGAT: Hierarchical Hyperbolic Knowledge Graph Attention Network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[39] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-Hoc Ranking with Kernel Pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Shinjuku, Tokyo, Japan) *(SIGIR '17)*. Association for Computing Machinery, New York, NY, USA.

[40] Liu Yang, Qingyao Ai, J. Guo, and W. Croft. 2016. aNMM: Ranking Short Answer Texts with Attention-Based Neural Matching Model. *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (2016).

[41] Justin Zobel and Alistair Moffat. 2006. Inverted Files for Text Search Engines. *ACM Comput. Surv.* 38, 2 (July 2006), 6–es.

A BROADER IMPACT

ANTHEM has the potential to have a large impact on product search and discovery. A vast majority of customers across countries start their shopping journey on e-commerce websites via a search functionality. Given the several millions of customers who interact with these systems, any improvements in performance of these systems (however small they are) has a large impact on the user base. Our work is aimed at practitioners and researchers in the broader data mining and machine learning communities who work in the domain of representation learning, particularly learning in the presence of hierarchical information.

Current systems model customer behaviors to provide more contextual information to improve search results. However, this customer information is both sensitive in nature and also a substantial source of bias [19]. In ANTHEM, we aim to provide a possible alternative which considers possible intents and statistically infers the right intent through historical purchases. Additionally, we design our model in a joint learning framework so that it conforms to existing architectures for easier deployment and is applicable to additional problems such as web search and semantic matching.

B COMPUTATIONAL COMPLEXITY

To analyze ANTHEM's potential of deployment in an industrial setting, we need to study the computational both in terms of its training and inference times. The model's parameter study, training time and inference run-time are provided in Table 5. To maintain a fair comparison, we do not include the overheads involved in the inference process such as loading the model and the request processing time of servers. We observe that the run-time of our models ANTHEM and E-ANTHEM are slightly higher than previous methods. The reason for this is the use of intersection and union which are quadratic operations. However, we note that $length(query) << length(answers)$ in product search engines, thus, the added complexity does not affect the runtime significantly. The difference in runtime is $\sim$ 10 seconds for 657K validation samples. Additionally, we also report a much lower number of model parameters in our models compared to the best performing baseline, i.e., BERT. This implies a lower training period (an advantage of $\sim 10,000$ seconds) which is beneficial to product search due to the dynamic nature and large-scale of product catalogues. Thus, we conclude that the slight increase in computational complexity is a fair trade-off for product search production systems given the lower number of model parameters (implying a lower training period) and additional interpretability of our models.

C QUERY-MATCHING ALGORITHM

Algorithm 2 provides the pseudo-code for training ANTHEM for the task of query matching. The algorithm utilizes cosine similarity to match the queries and return a probabilistic similarity measure for optimization (gradient back-propagation) using cross-entropy loss. The cosine similarity between query representations (set of hyperboloids) QS_q and QS_r is calculated as:

$$\hat{y} = \mu\left(\frac{\|QS_{q_i}.QS_{r_j}\|}{\|QS_{q_i}\|\|QS_{r_j}\|}\right) \forall\, QS_{q_i} \in QS_q, QS_{r_j} \in QS_r \qquad (14)$$

where $\mu(.)$ and $\|.\|$ represent the mean and Euclidean norm functions, respectively.

Table 5: Comparative analysis of computational complexity. q and a are the number of character trigrams in query and product sequences. The four final columns present the Training time taken per epoch (T) and Inference time per sample (I) of our model on different search datasets. The number of training and testing samples are given in Table 1. 'msec' stands for milliseconds.

Model	No. of Model Parameters	E-commerce Product Search		Public E-commerce Search Relevance	
		T(sec)	I(msec)	T(sec)	I(msec)
ARC-II	1,742,793	564	104	2.4	0.4
KNRM	1,667,522	540	100	2.3	0.4
DRMM	5,002,823	1,620	280	6.9	1.2
aNMM	9,037,903	2,927	338	12.5	1.4
Match Pyramid	1,660,361	538	100	2.3	0.4
C-DSSM	3,720,066	1,205	310	5.1	1.3
DUET	5,379,580	1,742	300	7.4	1.3
MV-LSTM	5,283,381	1,711	296	7.3	1.3
BERT	109,483,778	12,042	334	51.5	1.4
E-ANTHEM	6,374,554	2,064	332	8.8	1.4
ANTHEM	6,374,960	2,064	332	8.8	1.4

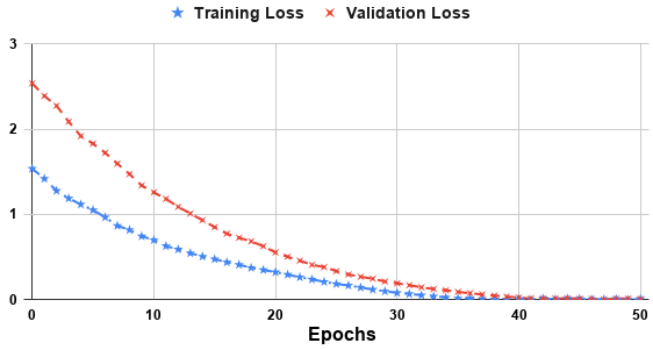
Algorithm 2: ANTHEM training for Query Matching

Data: Training data $D = (q, r \in Q, y \in \{1, ..., y_c\})$;
Output: Predictor X_ϕ;

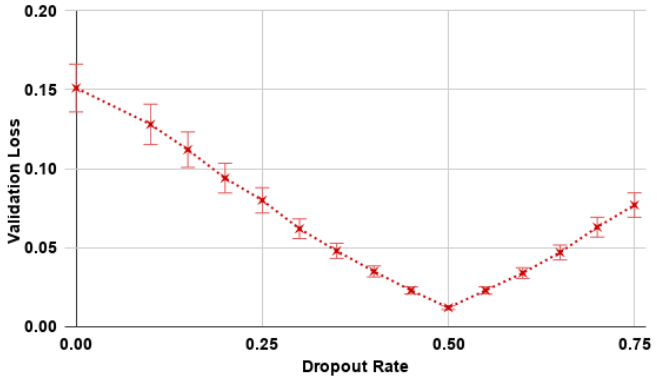
Initialize model parameters ϕ;
while *not converged* **do**
  $l = 0$; # Initialize loss
  for $\{(q, r, y) \in D\}$ **do**
    $q \leftarrow Embedding_\phi(q)$;
    $r \leftarrow Embedding_\phi(r)$;
    # Encode query q
    $H_q = f_{hyp}(q)$; using Eq. (1)
    $H_\cap = \{H_i \cap H_j \forall i, j : 1 \rightarrow n; H_i, H_j \in H_q$; via Eq. (6)
    $QS_q = f_{att}(H_q \cup H_\cap)$ via Eq. (9)
    # Encode query r
    $H_r = f_{hyp}(r)$;
    $H_\cap = \{H_i \cap H_j \forall i, j : 1 \rightarrow n; H_i, H_j \in H_r\}$;
    $QS_r = f_{att}(H_r \cup H_\cap)$
    # Calculate distance and update Loss l
    $\hat{y} = \mu\left(\frac{\|QS_{q_i}.QS_{r_j}\|}{\|QS_{q_i}\|\|QS_{r_j}\|}\right) \forall QS_{q_i} \in QS_q, QS_{r_j} \in QS_r$
    $l = l + L(\hat{y}, y)$; via Eq. (13)
    # Update ϕ with back-propagation
    $\phi \leftarrow \phi - \nabla_\phi l$;
  end
end
return X_ϕ

D SENSITIVITY TO HYPER-PARAMETERS

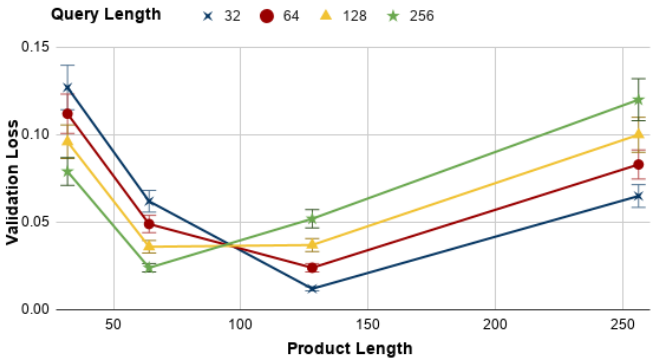
In this section, we study the sensitivity of our model with respect to the hyper-parameters. First, we analyze the convergence of our model across epochs and then we proceed to analyze the loss with



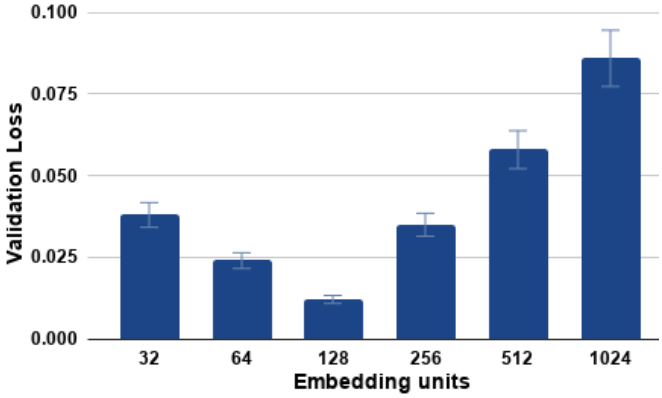
(a) Training and Validation loss across epochs (lower is better).



(b) Sensitivity of loss to dropout rate (lower is better).



(c) Sensitivity of loss to query and product length (lower is better).



(d) Sensitivity of loss to embedding dimensions (lower is better).

Figure 8: Illustration of parameter sensitivity of the proposed ANTHEM model.

varying dropout rates and lengths of query/product titles. Here, the length refers to number of character trigrams in the query or product. The results are presented in Figure 8.

In Figure 8a, we observe that ANTHEM is able to converge in under 50 epochs with Riemannian Adam optimizer (details provided in Section 3.4). Figure 8b illustrates the advantages of using dropout for efficient convergence and avoid overfitting. We observe that a dropout rate of 0.5 results in the most optimal solution and, hence, we adopt that in ANTHEM. In addition to this, we also need to find the most optimal query and product length. The model's complexity directly depends on these lengths. Hence, an optimal solution will significantly affect the training time. From Figure 8c, we observe that a query length of 32 with a product length of 128, provides the most optimal result. However, due to computational constraints, i.e., GPU VRAM < 8GB, we utilize a maximum query length of 28 and product length of 128. Finally, in the case of embedding units, we observe from Figure 8d that 128 is the optimal number of dimensions for least loss, and hence we utilize that for our model.

The final hyper-parameters adopted for E-ANTHEM and ANTHEM are query length of 28, product length of 512 and dropout rate of 0.5. For the baselines, the query length is 128 and product length is 512. The dropout rate for ARC-II, KNRM, DUET, DRMM, and aNMM is 0.2 and for MatchPyramid, C-DSSM, MV-LSTM, and BERT it is 0.5. The above hyper-parameters for the baselines are determined after extensive experimentation for the best performance.

E NUMBER OF GPUS

Figure 9 shows the dependence of training time on the number of GPUs, which depicts the feasibility of ANTHEM's parallelization. We notice that ANTHEM has a lower training time than the best performing comparison baseline, BERT.

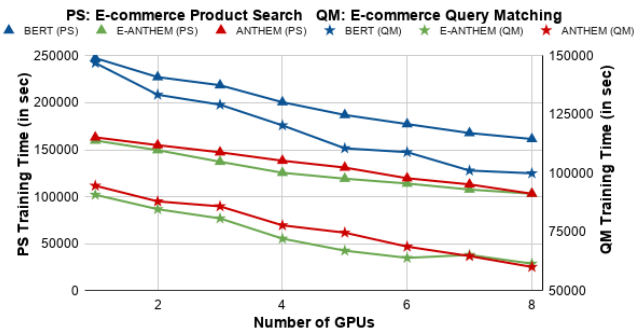


Figure 9: Total training time taken by ANTHEM and the best baseline (BERT) model using different number of GPUs (lower is better).