Multimodal Representation Learning for Textual Reasoning over Knowledge Graphs

Nurendra Choudhary

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science & Applications

Chandan K. Reddy, Chair
Chang Tien Lu
Lifu Huang
Karthik Subbian
Nikhil Rao

April 21, 2023
Falls Church, Virginia

Keywords: Multimodal representation learning, knowledge graphs, hyperbolic space, text reasoning, self-supervised, geometric search, product search, query representation.

Multimodal Representation Learning for Textual Reasoning over Knowledge Graphs

Nurendra Choudhary

(ABSTRACT)

Knowledge graphs (KGs) store relational information in a flexible triplet schema and have become ubiquitous for information storage in domains such as web search, e-commerce, social networks, and biology. Retrieval of information from KGs is generally achieved through logical reasoning, but this process can be computationally expensive and has limited performance due to the large size and complexity of relationships within the KGs. Furthermore, to extend the usage of KGs to non-expert users, retrieval over them cannot solely rely on logical reasoning but also needs to consider text-based search. This creates a need for multi-modal representations that capture both the semantic and structural features from the KGs.

The primary objective of the proposed work is to extend the accessibility of KGs to non-expert users/institutions by enabling them to utilize non-technical textual queries to search over the vast amount of information stored in KGs. To achieve this objective, the research aims to solve four limitations: (i) develop a framework for logical reasoning over KGs that can learn representations to capture hierarchical dependencies between entities, (ii) design an architecture that can effectively learn the logic flow of queries from natural language text, (iii) create a multi-modal architecture that can capture inherent semantic and structural features from the entities and KGs, respectively, and (iv) introduce a novel hyperbolic learning framework to enable the scalability of hyperbolic neural networks over large graphs using meta-learning.

The proposed work is distinct from current research because it models the logical flow of textual queries in hyperbolic space and uses it to perform complex reasoning over large KGs. The models developed in this work are evaluated on both the standard research setting of logical reasoning, as well as, real-world scenarios of query matching and search, specifically, in the e-commerce domain.

In summary, the proposed work aims to extend the accessibility of KGs to non-expert users by enabling them to use non-technical textual queries to search vast amounts of information stored in KGs. To achieve this objective, the work proposes the use of multi-modal representations that capture both semantic and structural features from the KGs, and a novel hyperbolic learning framework to enable scalability of hyperbolic neural networks over large graphs. The work also models the logical flow of textual queries in hyperbolic space to perform complex reasoning over large KGs. The models developed in this work are evaluated on both the standard research setting of logical reasoning and real-world scenarios in the e-commerce domain.

Multimodal Representation Learning for Textual Reasoning over Knowledge Graphs

Nurendra Choudhary

(GENERAL AUDIENCE ABSTRACT)

Knowledge graphs (KGs) are databases that store information in a way that allows computers to easily identify relationships between different pieces of data. They are widely used in domains such as web search, e-commerce, social networks, and biology. However, retrieving information from KGs can be computationally expensive, and relying solely on logical reasoning can limit their accessibility to non-expert users. This is where the proposed work comes in. The primary objective is to make KGs more accessible to non-experts by enabling them to use natural language queries to search the vast amounts of information stored in KGs. To achieve this objective, the research aims to address four limitations. Firstly, a framework for logical reasoning over KGs that can learn representations to capture hierarchical dependencies between entities is developed. Secondly, an architecture is designed that can effectively learn the logic flow of queries from natural language text. Thirdly, a multimodal architecture is created that can capture inherent semantic and structural features from the entities and KGs, respectively. Finally, a novel hyperbolic learning framework is introduced to enable the scalability of hyperbolic neural networks over large graphs using meta-learning. The proposed work is unique because it models the logical flow of textual queries in hyperbolic space and uses it to perform complex reasoning over large KGs. The models developed in this work are evaluated on both the standard research setting of logical reasoning, as well as, real-world scenarios of query matching and search, specifically, in the e-commerce domain.

In summary, the proposed work aims to make KGs more accessible to non-experts by enabling them to use natural language queries to search vast amounts of information stored in KGs. To achieve this objective, the work proposes the use of multi-modal representations that capture both semantic and structural features from the KGs, and a novel hyperbolic learning framework to enable scalability of hyperbolic neural networks over large graphs. The work also models the logical flow of textual queries in hyperbolic space to perform complex reasoning over large KGs. The results of this work have significant implications for the field of information retrieval, as it provides a more efficient and accessible way to retrieve information from KGs. Additionally, the multi-modal approach taken in this work has potential applications in other areas of machine learning, such as image recognition and natural language processing. The work also contributes to the development of hyperbolic geometry as a tool for modeling complex networks, which has implications for fields such as network science and social network analysis. Overall, this work represents an important step towards making the vast amounts of information stored in KGs more accessible and useful to a wider audience.

Dedication

To my family and friends, your unwavering love and support has made this achievement possible, and this Thesis is dedicated to you.

Acknowledgments

I am deeply grateful for the support and guidance of my advisor, Dr. Chandan Reddy, throughout my PhD journey. Dr. Reddy's unwavering dedication to his students and his tireless efforts to help me achieve my research goals have been invaluable to me. His critiques and feedback helped me to become a better researcher, and his mentorship will continue to have a positive impact on my future career.

I also want to express my thanks to the members of my dissertation committee: Dr. Chang Tien Lu, Dr. Lifu Huang, Dr. Nikhil Rao, and Dr. Karthik Subbian. Their expertise and insights were instrumental in shaping the direction of my research, and their feedback helped me to improve the quality of my work. I am grateful for their time, energy, and support.

I would like to acknowledge my collaborators, including Dr. Sumeet Katariya, Dr. Edward W. Huang, Mehrdad Khatir, Dr. Charu Aggarwal, Dr. Sutanay Choudhury, and Dr. Khusbu Agarwal. Their contributions to my research were crucial, and I appreciate the many hours we spent discussing ideas, analyzing data, and writing papers. I am grateful for their expertise, friendship, and support.

The Sanghani Center for Artificial Intelligence and Data Analytics provided an excellent research environment, and I want to thank the lab members for their support and friendship. Special thanks go to Dr. Khoa Doan, Mehrdad Khatir, Akshita Jha, Sindhu Tipirneni, Dr. Ping Wang, Dr. Tian Shi, Dr. Nikhil Muralidhar, and Dr. Sathappan Muthiah for their support, encouragement, and collaboration. Their insights and expertise have been invaluable to me, and I am grateful for the many conversations we had in the lab.

I also want to express my thanks to my friends, including Dr. Abhirup Dikshit, Sakshi Mishra, Kuldeep Dixit, Dr. Sukrit Venkatagiri, Dr. Debanjan Datta, and Dr. Subhodip Biswas. Their friendship, encouragement, and support were important to me, especially during the challenging times of my PhD journey. I am grateful for the memories we shared and the lessons I learned from them.

I cannot express my gratitude enough to my parents, who provided me with unconditional love and support throughout my life. Their sacrifices and encouragement have been essential to my success, and I will always be grateful for their unwavering belief in me.

Finally, I want to acknowledge my parents, who were always there for me during the past several years. Their love, support, and understanding helped me to navigate the challenges of my PhD journey, and I am grateful for their unwavering encouragement. My accomplishment is not mine alone, but it belongs to them as well. Thank you for being my rock and my inspiration

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Knowledge Graphs (KGs) are structured heterogeneous graphs where information is organized as triplets of entity pair and the relation between them. This organization provides a fluid schema with applications in several domains including e-commerce [38], web ontologies [7, 9], and medical research [74, 110]. However, retrieval of useful information from these KGs is a challenging task due to their huge size and relation complexity. Search over them through standard techniques such as graph traversal or heuristic A^* search algorithm still requires a logarithmic time complexity. But, in practical applications such as web search and e-commerce, the results need to be retrieved in real-time. Research into computationally efficient retrieval [26, 55, 104] shows that learning entity and relation representations and using logical first-order queries such as translation, intersection, and union can significantly improve time complexity. However, logical querying requires the user to be aware of the internal search database, as well as, have a certain level of expertise in forming the correct logical queries. Thus, this research aims to make the knowledge graphs accessible to wider community of users with different levels of expertise. To this end, the work targets the following broader research problems; (i) Effective logical reasoning over knowledge graphs, (ii) Retrieval over knowledge graphs through inferred logical reasoning using natural language queries, (iii) Integrating the framework with multi-modal semantic and structural information, and (iv) Improving scalability over large graphs. The research issues and contributions in each of the given sub-problems are provided in the subsequent sections.

1.1 Research Issues

This research aims to extend the accessibility of knowledge graphs to non-expert users/institutions who will be able to utilize non-technical textual queries to search over information organized in knowledge graphs. The major research issues are stated as follows;

- **Effective logical reasoning over knowledge graphs:** Although Euclidean spaces have proven to be effective for representation learning in various domains [5], several hierarchical datasets (including graph data) in the fields of network sciences and E-commerce taxonomies demonstrate a latent non-Euclidean anatomy [13]. De Sa et al. [32] shows that Euclidean spaces with unbounded number of dimensions cannot contain arbitrary tree structures. The task is trivial in a hyperbolic space with only 2-

dimensions where the exponential growth of distances is proportional to the growth in number of nodes with depth. The introduction of hyperbolic algebraic operations [45] such as Möbius addition, Möbius scalar multiplication and Exponential/Logarithmic maps, have led to the proliferation of hyperbolic neural networks such as Hyperbolic-GCN (H-GCN) [17] and Hyperbolic Attention (HAT) networks [50]. These frameworks leverage the hyperbolic anatomy of hierarchical datasets and show a significant performance boost compared to their Euclidean counterparts. To the best of our knowledge, there is no existing work that (i) utilizes dynamic computational graphs on the hyperbolic space, and (ii) applies complex hyperbolic geometries such as hyperboloids for representation learning. Additionally, the static computational graphs of H-GCN and HAT limit their learning capability to a single problem, generally, multi-hop (translation) reasoning. This severely limits their applicability to representation learning on KGs since translations can only utilize single entities. More complex intersections and unions not only use more entities, but are also more representative of real-world KG queries. While solving union and intersection queries is more challenging, they enable better representation learning [5]. Traversing over the entities in KGs facilitates an intuitive way of constructing a query-reasoning proxy task that enables representation learning of entities and relations. These representations, in a self-supervised framework, can further provide enriched features in downstream tasks (such as anomaly detection), thus alleviating the issue of data scarcity.

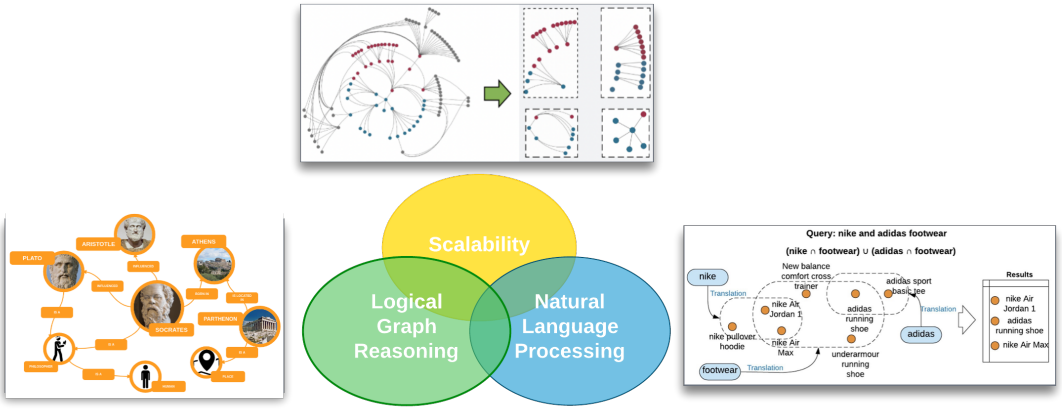
- **Retrieval through reasoning over knowledge graphs using natural language queries:** Current search frameworks include two major modules for retrieving the product matches for a given input query [96]; (i) a matching phase that generates a set of items deemed appropriate to the query, and (ii) a ranking phase that ranks these items in a certain order of suitability. Traditional approaches for matching [81, 163] 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 [63, 96] learn a joint query-answer 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 [8] while the deep learning encoders capture semantic information from both the query and answers. However, these approaches are limited due to the following challenges; (i) **Hierarchical structure**: Existing methods do not leverage the inherent hierarchy present in the answer knowledge graphs. This motivates the need for using hyperbolic spaces that better conform to the latent anatomy of knowledge graphs compared to their Euclidean counterparts [45]; (ii) **Dynamic query space**: Current matching approaches utilize a fixed threshold (top-K retrieval) to return answers in

the match set. However, general queries like *men shoes* should match onto a larger portion of the knowledge graphs 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 answers for general queries; and (iii) **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 queries are, generally, composed of independent tokens with hierarchical connections. Thus, query composition depends upon capturing the complex hierarchical intersection/union between answer tokens and their individual semantic information, e.g., $P(nike\ adidas) = P(nike \cup adidas)P(nike)P(adidas)$.

- **Integrating the framework with multi-modal semantic and structural information:** In this problem, we aim to create a unified graph representation learning methodology that tackles the following challenges; (i) *Leveraging global graph structure:* Existing GNN frameworks aggregate information only from a local neighborhood of the graph and do not possess the ability to aggregate global graph structures. Indeed, when attempting to combine information from the entire graph, existing methods suffer from over-smoothness [97]. Moreover, the size of modern graph datasets renders aggregating information from the full graph infeasible; (ii) *Incorporating hierarchical structures:* Most of the real-world graphs have inherent hierarchies, which are best represented in a hyperbolic space (rather than the traditional Euclidean space). However, existing hyperbolic GNNs [17, 45] do not leverage the full graph when aggregating information due to both mathematical and computational challenges; (iii) *Integrating textual (semantic) content:* Previous methods for integrating semantic information of the nodes are relatively ad-hoc in nature. For example, they initialize their node representations with text embeddings for message aggregation in the GNNs [159]. Such methods fix the semantic features and do not allow the framework to learn task-specific embeddings directly from the nodes' original content; and (iv) *Robustness to noise:* Real-world graphs are susceptible to noise and hence require robust graph representation learning mechanisms, especially in the presence of multiple forms of data (i.e., graph structure and textual content). Previous approaches do not leverage the complementary nature of graphs and text to improve robustness to noise in both of these modalities.

- **Improving scalability over large graphs:** In Euclidean GNNs, the nodes' or edges' information is only dependent on the immediate neighborhood, and thus, they have benefited from node-level extensions of traditional sampling methods [20, 41] to improve their parallelism and scalability on large datasets. Hyperbolic neural networks (HNNs), on the other hand, have not been able to adopt these advances in scalability. Traditional HNN models such as HGCN [17], HAT [50], and HypE [26] have been designed for experimental datasets, and hence, they are limited in their ability to learn inductive biases, i.e., local subgraph information that can generalize over the entire graph. The HNN formulations [45] depend on the global origin (root node) for several

transformation operations (Möbius addition, Möbius multiplication and others), and thus, hyperbolic representation of a node becomes erratic in the absence of origin. However, HNNs have shown impressive performance on several research domains including recommendation systems [117], e-commerce [28], natural language processing [35], and knowledge graphs [18, 26]. Their ability to efficiently encode exponentially increasing hierarchical graph datasets with lower distortion than the Euclidean space is quite beneficial. Thus, it is important that we scale HNNs to attain the gains in performance on large graph datasets too. To achieve this, we introduce a novel method, Hyperbolic GRAph Meta Learner (H-GRAM), that utilizes meta-learning to learn information from local subgraphs for HNNs and transfer it for faster learning on a disjoint set of nodes, edges and labels contained in the larger graph. As a consequence of meta-learning, H-GRAM also achieves several desirable benefits that extends HNNs' applicability including the ability to transfer information on new graphs (inductive learning), elimination of over-smoothing, and few-shot learning.



CONTRIBUTIONS			
Effective Logical Reasoning (HyPE)	**Reasoning-based Retrieval (ANTHEM)**	**Multi-modal Representations (TESH-GCN)**	**Meta-Learning based Scalability (H-GRAM)**
Proposed a hyperbolic network to reason over Knowledge graphs. 1. Captures spatial features of entities and relations. 2. Captures hierarchical features of entities and relations. 3. Simultaneously handles translation, intersection and union. 4. Self-supervised over the graph and does not require manual labeling.	Proposed a retrieval model based to search over HyPE representations using natural language queries. 1. HypE embeddings capture the spatial and hierarchical features. 2. Query model learns logical operations from NL queries using a modified attention mechanism. 3. The model is a modular twin-tower architecture to support adoption in existing platforms.	Proposed a representation model that captures multi-modal semantic and structural information. 1. The hyperbolic model learns metapaths from global structures. 2. Integrates the semantic content of the node in relation learning graph pipeline. 3. Improves robustness to text and graph noise in the datasets and promotes model explanability.	Proposed a meta-learning model to improve scalability of hyperbolic networks over large graphs. 1. Introduces the theory to prove that local neighborhoods provides the majority of a node's information. 2. H-GRAM learns meta information form local subgraphs and generalizes it over the entire graph. 3. Solves HNN challenges in HNNs including inductive learning, elimination of over-smoothing.
Logical Reasoning	**Multi-modal Graph and Text Representations**		**Scalability**

Figure 1.1: Research problems and contributions of this Thesis.

1.2 Contributions

The major proposed research contributions, shown in Figure 1.1 can be stated as follows;

- **Effective logical reasoning over knowledge graphs:** (i) Formulate the KG representation learning problem as a self-supervised query reasoning problem to leverage PFOE queries; (ii) Introduce Hyperboloid Embeddings (HypE), a self-supervised dynamic representation learning framework that learns hyperboloid representations of KG units in a Poincaré hyperball. This is motivated by the need for non-Euclidean geometries; (iii) Perform an extensive set of empirical studies across diverse set of real-world datasets to evaluate the performance of HypE against several state-of-the-art baseline methods on the downstream task of Anomaly Detection.; and (iv) Visualize the HypE embeddings to clearly interpret and comprehend the representation space.

- **Retrieval through reasoning over knowledge graphs using natural language queries:** (i) A novel 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; (ii) A mechanism that utilizes attention units' activation to understand the internal working of ANTHEM and explain its product search mechanism on sample queries; (iii) 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; and (iv) 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.

- **Integrating the framework with multi-modal semantic and structural information from entities and knowledge graphs, respectively:** (i) We introduce Text Enriched Sparse Hyperbolic Graph Convolution Network (TESH-GCN), which utilizes semantic signals from input nodes to extract the local neighborhood and global graph features from the adjacency tensor of the entire graph to aid the prediction task; (ii) To enable the coordination between semantic signals and sparse adjacency tensor, we reformulate the hyperbolic graph convolution to a linear operation that is able to leverage the sparsity of adjacency tensors to reduce the number of model parameters, training and inference times (in practice, for a graph with 10^5 nodes and 10^{-4} sparsity this reduces the memory consumption from 80GB to 1MB). To the best of our knowledge, no other method has utilized the nodes' semantic signals to extract both local and global graph features; (iii) Our unique integration mechanism, not only captures both graph and text information in TESH-GCN, but also, provides robustness against noise in the individual modalities; and (iv) We conduct extensive experiments on a diverse set of graphs to compare the performance of our model against the state-of-

the-art approaches and also provide an explainability method to better understand the internal workings of our model.

- **Improving scalability over large graphs using meta-learning:** (i) We theoretically prove that HNNs rely on the nodes' local neighborhood for evidence in prediction, as well as, formulate HNNs to encode node-centric local subgraphs with root nodes as the local origin using the locality of tangent space transformations; (ii) We develop Hyperbolic GRAph Meta Learner (H-GRAM), a novel method that learns meta information (as meta gradients and label protonets) from local subgraphs and generalize it to new graphs with a disjoint set of nodes, edges and labels. Our experiments show that H-GRAM can be used to generalize information from subgraph partitions of large datasets, thus, enabling scalability in hyperbolic models; and (iii) Our analysis on a diverse set of datasets demonstrates that our meta-learning setup also solves several challenges in HNNs including inductive learning, elimination of over-smoothing and few-shot learning in several challenging scenarios.

1.3 Thesis Organization

The remainder of this research proposal is organized as follows; Chapter 2 introduces the self-supervised hyperboloid embeddings approach towards reasoning over knowledge graphs with hierarchical features. Chapter 3 describes the proposed ANTHEM model for learning logical operations from natural language queries and presents extensive quantitative and qualitative experimental results. Chapter 4 details the proposed TESH-GCN model that coherently learns semantic and structural information from knowledge graphs. Chapter 5 explains the H-GRAM learning framework that promotes scalability by meta-learning over local subgraphs and transferring the information over the entire graph structure. Chapter 6 concludes the Thesis and discusses the future directions.

Chapter 2

Self-Supervised Hyperboloid Representations from Logical Queries over Knowledge Graphs

Knowledge Graphs (KGs) are ubiquitous structures for information storage in several real-world applications such as web search, e-commerce, social networks, and biology. Querying KGs remains a foundational and challenging problem due to their size and complexity. Promising approaches to tackle this problem include embedding the KG units (e.g., entities and relations) in a Euclidean space such that the query embedding contains the information relevant to its results. These approaches, however, fail to capture the hierarchical nature and semantic information of the entities present in the graph. Additionally, most of these approaches only utilize multi-hop queries (that can be modeled by simple translation operations) to learn embeddings and ignore more complex operations such as intersection, and union of simpler queries. To tackle such complex operations, in this chapter, we formulate KG representation learning as a self-supervised logical query reasoning problem that utilizes translation, intersection and union queries over KGs. We propose Hyperboloid Embeddings (HypE), a novel self-supervised dynamic reasoning framework, that utilizes positive first-order existential queries on a KG to learn representations of its entities and relations as hyperboloids in a Poincaré ball. HypE models the positive first-order queries as geometrical translation, intersection, and union. For the problem of KG reasoning in real-world datasets, the proposed HypE model significantly outperforms the state-of-the art results. We also apply HypE to an anomaly detection task on a popular e-commerce website product taxonomy as well as hierarchically organized web articles and demonstrate significant performance improvements compared to existing baseline methods. Finally, we also visualize the learned HypE embeddings in a Poincaré ball to clearly interpret and comprehend the representation space.

2.1 Introduction

Knowledge Graphs (KGs) organize information as a set of entities connected by relations. Positive first-order existential (PFOE) queries such as translation, intersection, and union over these entities aid in effective information extraction from massive data (see Figure 2.1

for an example PFOE query). Efficient handling of such queries on KGs is of vital importance in a range of real-world application domains including search engines, dialogue systems, and recommendation models. However, the large size of KGs and high degrees of the nodes therein makes traversal for querying a computationally challenging or, in some cases, even an impossible task [132]. One way to resolve this issue is to learn representations for the KG units (entities and relations) in a latent (generally Euclidean) space such that algebraic or logical operations can be applied to extract relevant entities. Robust representation learning of KG units has several real-world applications including KG information extraction [55], entity classification [136], and anomaly detection [66].

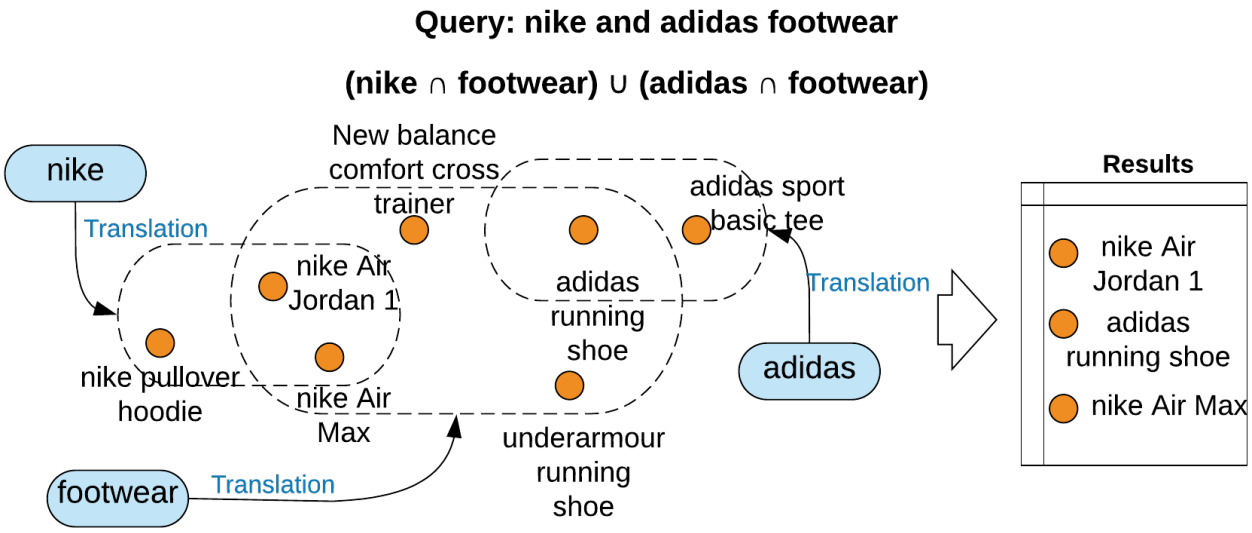
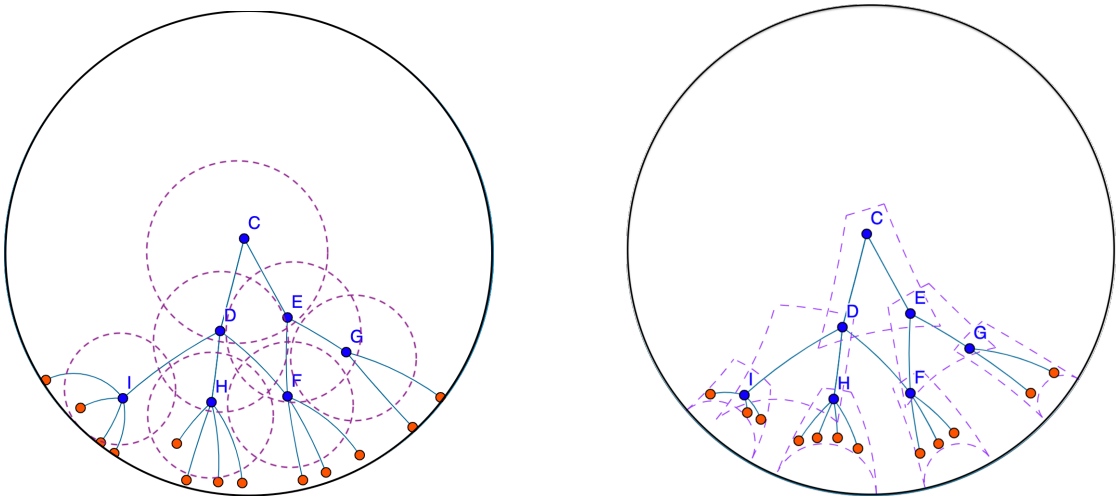


Figure 2.1: An example of PFOE querying in the E-commerce product network. The product space of `Adidas` and `Nike` intersects with `Footwear` to narrow the search space. A union over these spaces yields our final set of entity results.

Earlier approaches to representation learning in KGs model entities and relations as vectors in the Euclidean space [11, 95, 142]. This is suboptimal due to the constant size of a point's answer space which does not capture the variations induced by different queries. Specifically, broad queries (`Nike`) should intuitively cover a larger region of the answer space compared to specific queries (`Nike running shoes for men`). In the recently proposed Query2Box model [104], the authors demonstrated the effectiveness of complex geometries (such as hyper-rectangles) with varying offsets that control the size of an answer space according to a query's complexity. However, such architectures lack the ability to capture hierarchical information that is prevalent in many KGs. Furthermore, previous representation learning methods in heterogeneous graphs (or KGs) [44, 77, 79, 154] solely focus on one-hop or multi-hop reasoning over relations. Such frameworks enable static and optimized computational graphs, but lead to poor retrieval from complex intersection and union queries. Dynamic computational graphs, which are able to modify their network architecture with a switch mechanism (discussed in Section 2.3.5) can significantly alleviate this problem.

Although Euclidean spaces have proven to be effective for representation learning in various domains [5], several hierarchical datasets (including graph data) in the fields of network sciences and E-commerce taxonomies demonstrate a latent non-Euclidean anatomy [13]. The introduction of hyperbolic algebraic operations [45] have led to the proliferation of hyperbolic neural networks such as Hyperbolic-GCN (H-GCN) [17] and Hyperbolic Attention (HAT) networks [50]. These frameworks leverage the hyperbolic anatomy of hierarchical datasets and show a significant performance boost compared to their Euclidean counterparts. To the best of our knowledge, there is no existing work that (i) utilizes dynamic computational graphs on the hyperbolic space, (ii) applies complex hyperbolic geometries such as hyperboloids for representation learning. Additionally, the static computational graphs of H-GCN and HAT limit their learning capability to a single problem, generally, multi-hop (translation) reasoning. This severely limits their applicability to representation learning on KGs since translations can only utilize single entities. **More complex intersections and unions not only use more entities, but are also more representative of real-world KG queries.** While solving union and intersection queries is more challenging, they enable better representation learning [5]. Traversing over the entities in KGs facilitates an intuitive way of constructing a query-reasoning proxy task (refer Section 2.4.3) that enables representation learning of entities and relations. These representations, in a self-supervised framework, can further provide enriched features in downstream tasks with smaller annotated datasets (such as anomaly detection), thus alleviating the issue of data scarcity.



(a) Hyperbolic vectors (b) Hyperboloids

Figure 2.2: Static vs Dynamic representation. (a) hyperbolic vectors have lower precision due to static thresholds over a center, depicted by the dotted circle. (b) Dotted hyperboloids encapsulate all its child entities because of dynamic sizes. The blue and orange circles are intermediate and leaf nodes.

Motivated by the effectiveness of self-supervised learning and the need for non-Euclidean geometries in KGs, we formulate KG representation learning as a self-supervised query rea-

soning problem. *We introduce Hyperboloid Embeddings (HypE), a self-supervised dynamic representation learning framework that utilizes PFOE queries to learn hyperboloid representations of KG units in a (non-Euclidean) Poincaré hyperball.* Hyperboloids, unlike vectors in hyperbolic spaces, allow us to use dynamic sizes for KG representations. For e.g., in Figure 2.2, we can notice that different entities contain different number of children, and, thus learning a static vector representation is suboptimal. Hyperboloids learn an additional spatial parameter, *limit* (described in Section 2.3.4), that can model the varying entity sizes. Moreover, the dynamic nature of its computational graphs allows HypE to utilize different network layers to learn different types of operations, namely, translation, intersection, and union; and process varying number of input units depending on the learning operation. Our empirical studies include learning representations from large-scale KGs in e-commerce, web pages (DBPedia), and other widely used Knowledge bases (such as Freebase and NELL995); and evaluating the representations on the downstream task of anomaly detection. The major contributions of this chapter are:

1. Formulate the KG representation learning problem as a self-supervised query reasoning problem to leverage PFOE queries.

2. Introduce Hyperboloid Embeddings (HypE), a self-supervised dynamic representation learning framework that learns hyperboloid representations of KG units in a Poincaré hyperball. This is motivated by the need for non-Euclidean geometries.

3. Perform an extensive set of empirical studies across diverse set of real-world datasets to evaluate the performance of HypE against several state-of-the-art baseline methods on the downstream task of Anomaly Detection.

4. Visualize the HypE embeddings to clearly interpret and comprehend the representation space.

The rest of the chapter is organized as follows: Section 2.2 describes the related background. Section 2.3 formulates the representation learning problem, explains the non-Euclidean algebraic operations, and the proposed HypE model. In Section 2.4, we describe the real-world datasets, state-of-the-art baselines and performance metrics used to evaluate the HypE model. We demonstrate the performance results along with the visualization of HypE's representations. Finally, Section 2.5 concludes the chapter.

2.2 Related Work

In this section, we review different geometries utilized for learning representations and earlier works that are adopted for reasoning over Knowledge Graphs.

Representation Geometries : Previous approaches to representation learning, in the context of KG, aim to learn latent representations for entities and relations. Translational frameworks [11, 94, 142] model relations using translation between entity pairs. This limits the models to only handle translation-based queries. Graph Query Embedding (GQE) [55] overcame this limitation and provided a technique for leveraging intersection queries as deep sets [150] over different queries. Furthermore, Box Lattices [127], EMQL [116] and Query2Box [104] proved the effectiveness of more complex geometries (hyper-rectangles) for queries. Word2Gauss [126] is a popular NLP technique that learns Gaussian embeddings for words. DNGE [121] utilizes a dynamic network to learn Gaussian embeddings for entities in a graph. These Gaussian representations cannot be intuitively extended to KGs because they are not closed under more complex PFOE queries (intersection or union of Gaussians does not yield a Gaussian). Furthermore, they rely on properties of the Euclidean space to learn representations, which are proven ineffective at capturing the prevalent hierarchical features of a KG [45].

Representation Learning on Graphs : One of the fundamental problems in KG is to aggregate the neighbor information of nodes while learning representations. Node embedding techniques such as Node2Vec [49] and DeepWalk [101] aggregate the neighbors' features by modeling the node's dependence on its neighbors. ChebNet [33] uses Chebyshev polynomials and filters node features in the graph Fourier domain. GCN [70] constrains the parameters of ChebNet to alleviate overfitting and shows improved performance. Graph-BERT [154] and MAGNN [44] provide a self-supervised learning model utilizing the tasks of masking and metapath aggregation, respectively. In another line of research, Miller et al. [83] utilizes non-parametric Bayesian frameworks for link prediction on social networks. Zhu [160] further improved the approach with a max-margin framework. KGAT [134] is another popular approach that utilizes attention networks over entities and relations with a TransR [76] loss function to learn representations for user recommendation. These methods rely on relational properties and thus are effective in handling translational problems such as multi-hop reasoning. However, they are ineffective at handling more complex PFOE queries such as intersection and union.

Other popular multi-hop graph networks such as Graph Attention Network (GAT) [124] and Graph Recurrent Network (GRN) [114] have previously shown impressive results in reasoning-based QA tasks. However, hyperbolic flavors of these networks, H-GNN [45], H-GCN [17, 18] and H-GAT [50] argue that hierarchical datasets follow the anatomy of hyperbolic space and show improved performance over their Euclidean counterparts. Nonetheless, these approaches are still limited by the constant answer space that does not consider the varying fluctuations of complex queries.

Self-supervised learning [37, 64, 86, 141] utilizes large unannotated datasets to learn representations that can be fine-tuned to other tasks that have relatively smaller amount of annotated data. Traversing over the entities in KGs facilitates an intuitive way of construct-

ing a query-reasoning proxy task (refer Section 2.4.3) that enables representation learning of entities and relations. These representations, in turn, are employed in downstream tasks with scarce datasets such as anomaly detection.

The proposed HypE model utilizes a self-supervised learning framework that leverages both simple and complex PFOE queries to learn hyperboloid (with varying limits) representations of KG units in a Poincaré ball to efficiently capture hierarchical information.

2.3 Proposed Framework

In this section, we first provide the standard method of querying knowledge graphs. Then, we set up the problem and describe the details of our model that learns representations of entities and relations from reasoning queries over Knowledge Graphs (KG).

2.3.1 Querying Knowledge Graphs

KGs contain two primary units, namely, entities and relations. Entities are the basic information units that connect to each other by relation units. Heterogeneous graphs [135, 152] can be considered as a special case of KGs where the relations serve as hierarchical connections with no inherent information. PFOE queries of translation (t), intersection ($\cap$) and union ($\cup$) serve as the primary means of querying these KGs. Translation queries utilize an entity e and a relation r to retrieve all entities that are connected to e through r. An equivalent example of translation query for heterogeneous graphs is to retrieve all children of a node e connected by a certain edge type r. Intersection and Union operate over multiple entities $E = \{e_1, .., e_n\}$ and correspondingly retrieve the set of all entities that are connected to all $e_i \in E$ and any $e_i \in E$. For heterogeneous graphs, the equivalent is to retrieve nodes connected to all nodes $e_i \in E$ and any $e_i \in E$. An example of PFOE querying is given in Figure 2.3. The widely studied problem of multi-hop traversal [44] is a more specific case of translation queries, where multiple queries are chained in a series.

2.3.2 Problem Setup

We denote $KG = (E, R)$ as a set of entities $e_i \in E$ and relations $r_{ij} \in R \quad : \quad e_i \rightarrow e_j$ as Boolean functions that indicate whether a directed relation r_{ij} holds between e_i and e_j. Intersection ($\cap$) and Union ($\cup$) are positive first-order existential (PFOE) operations defined on a set of queries $q_1, q_2, ..., q_n \in Q$:

$$q_\cap[Q] = V_\cap \subseteq E \;\; \exists \;\; e_1, e_2, ..., e_k : q_1 \cap q_2 \cap q_3 ... \cap q_n$$
$$q_\cup[Q] = V_\cup \subseteq E \;\; \exists \;\; e_1, e_2, ..., e_k : q_1 \cup q_2 \cup q_3 ... \cup q_n$$

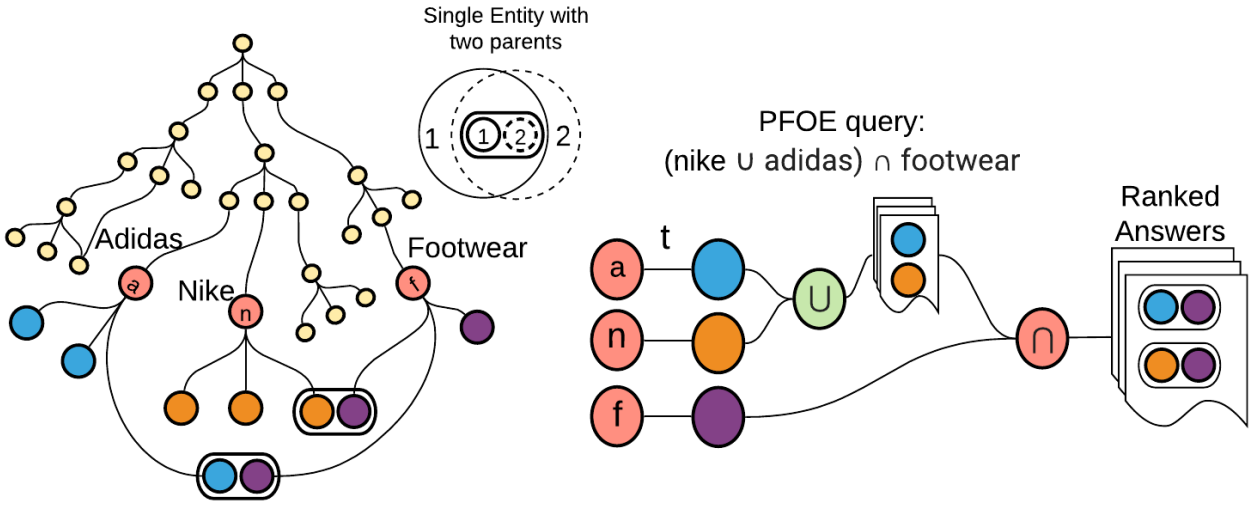


Figure 2.3: A simple first-order query over a Product Graph. The query `Nike and Adidas footwear` can be expressed as the union of the Nike and Adidas nodes intersected with the node corresponding to footwear in the product catalog.

where $(q_\cap, V_\cap)$ and $(q_\cup, V_\cup)$ is the query space and resultant entity set after the intersection and union operations over query set Q, respectively[1]. Given a dataset of logical queries over a KG, the goal, here, is to learn hyperboloid (in a Poincaré ball) representations for its entities $e_i \in \mathbb{R}^{2d}$ and relations $r_{ij} \in \mathbb{R}^{2d}$, where d is a hyper-parameter that defines the dimension of the Poincaré ball.

2.3.3 Manifold Transformation Layer

Hierarchical structures intuitively demonstrate the latent characteristics of a hyperbolic space [45]. Thus, we utilize the Poincaré ball [14] to model our representations.

Transformation from Euclidean Space

The transformation from Euclidean to hyperbolic space $(\mathbb{H}^n, g^{\mathbb{H}})$, given in [45], is defined by the manifold $\mathbb{H}^n = \{x \in \mathbb{R}^n : \|x\| < 1\}$ with the Reimannian metric , $g^{\mathbb{H}}$, where:

$$g_x^{\mathbb{H}} = \lambda_x^2 g^{\mathbb{E}} \quad \text{where } \lambda_x \coloneqq \frac{2}{1 - \|x\|^2} \tag{2.1}$$

$g^{\mathbb{E}} = \mathbf{I}_n$ being 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 set to a

[1] e.g., if $(q_i, \{e_a, e_b\})$ and $(q_j, \{e_b, e_c\})$ are the corresponding query space and resultant entity sets, then $q_\cap[\{q_i, q_j\}] = \{e_b\}$ and $q_\cup[\{q_i, q_j\}] = \{e_a, e_b, e_c\}$.

conventional curvature of -1. Eq. (2.1) allows us to convert a Euclidean metric to hyperbolic. Thus, the distance between points $x, y \in \mathbb{H}^n$ is derived as:

$$d_{\mathbb{H}}(x, y) = \cosh^{-1}\left(1 + 2\frac{\|x - y\|^2}{(1 - \|x\|^2)\,(1 - \|y\|^2)}\right) \tag{2.2}$$

Gyrovector Spaces

Algebraic operations such as addition and scalar product which are straightforward in the Euclidean space cannot be directly applied in hyperbolic space. Gyrovector spaces allow for the formalization of these operations in hyperbolic space.

Ganea et al. [45] provide the gyrovector operations relevant to training neural networks. The operations for Poincaré ball of radius c are Möbius addition ($\oplus_c$), Möbius subtraction ($\ominus_c$), exponential map ($\exp_x^c$), logarithmic map ($\log_x^c$) and Möbius scalar product ($\odot_c$).

$$\begin{aligned}
x \oplus_c y &:= \frac{(1 + 2c\langle x, y\rangle + c\|y\|^2)\,x + (1 - c\|x\|^2)\,y}{1 + 2c\langle x, y\rangle + c^2\|x\|^2\|y\|^2} \\
x \ominus_c y &:= x \oplus_c -y \\
\exp_x^c(v) &:= x \oplus_c \left(\tanh\left(\sqrt{c}\frac{\lambda_x^c\|v\|}{2}\right)\frac{v}{\sqrt{c}\|v\|}\right) \\
\log_x^c(y) &:= \frac{2}{\sqrt{c}\lambda_x^c}\tanh^{-1}\left(\sqrt{c}\|-x \oplus_c y\|\right)\frac{-x \oplus_c y}{\|-x \oplus_c y\|} \\
r \odot_c x &:= \exp_0^c(r log_0^c(x)),\ \forall r \in \mathbb{R}, x \in \mathbb{H}_c^n
\end{aligned}$$

Here, $:=$ denotes assignment operation for Möbius operations. Also, the norm of x, y can subsume the scaling factor c. Hence, in HypE, training can be done with a constant c or trainable c. We empirically validate this assumption in our experiments (Section 2.4.4). Figure 2.4 shows an example of the manifold transformation from Euclidean space to a Poincaré ball of unit radius. HypE extends the operations to handle complex geometries, explained in Section 2.3.4.

2.3.4 Dynamic Reasoning Framework : HypE

We aim to learn hyperboloid (made of two parallel pairs of arc-aligned horocycles) embeddings for all the entities and relations in the KG. An arc-aligned horocyle (Figure 2.5(a)) is a partial circle that is parallel to the diameter of a Poincaré ball and orthogonally intersects its boundaries at two points. A hyperboloid embedding (see Figure 2.5(b)) $e = (\text{cen}(e), \lim(e)) \in \mathbb{R}^{2d}$ is characterized by:

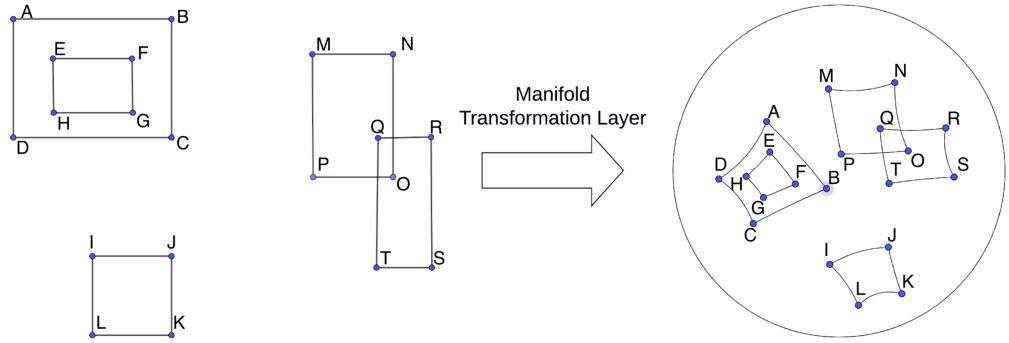
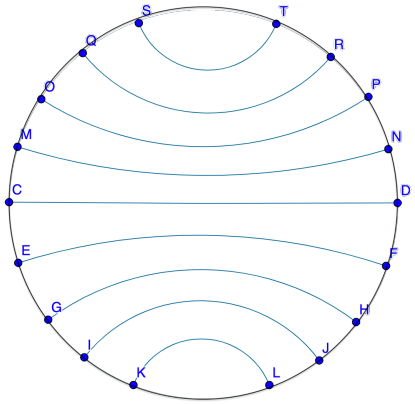
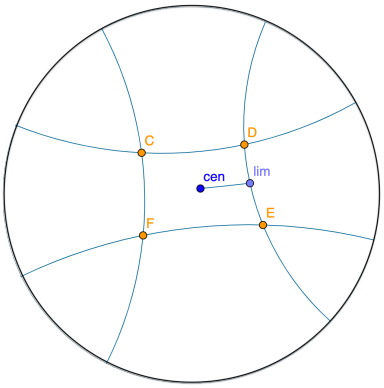


Figure 2.4: Manifold Transformation of Euclidean geometries (rectangles) to a Poincaré ball (horocycle enclosures).



(a) Horocycles in a Poincaré ball.



(b) CDFE is the hyperboloid.

Figure 2.5: Horocycles and hyperboloids in a Poincaré ball. The hyperboloid is composed of two parallel pairs of arc-aligned horocycles.

$$H_e \equiv \{v \in \mathbb{R}^d : \text{cen}(e) \ominus_c \lim(e) \leq v \leq \text{cen}(e) \oplus_c \lim(e)\}$$

where $\equiv$ denotes strict equivalence and $\leq$ is element-wise inequality and $\text{cen}(t), \lim(t) \in \mathbb{R}^d$ are center of the hyperboloid and positive border limit ($\lim(t) \geq 0$) of the enclosing horocycle from the center, respectively. The overview of the architecture is given in Figure 2.6. From the KG, we derive the following types of directed edge relations to build our dynamic computational graph for learning embeddings.

Distance between hyperboloid and entity point (d): Given a query hyperboloid $q \in \mathbb{R}^{2d}$ and entity center $v \in \mathbb{R}^d$, the distance between them is defined as:

$$\begin{aligned} d_{hyp}(v, q) &= d_{out}(v, q) \oplus_c \gamma d_{in}(v, q) \\ d_{out}(v, q) &= \|Max(d_{\mathbb{H}}(v, q_{max}), 0) + Max(d_{\mathbb{H}}(q_{min}, v), 0)\|_1 \\ d_{in}(v, q) &= \|cen(q) \ominus_c Min(q_{max}, Max(q_{min}, v))\|_1 \\ q_{min} &= cen(q) \ominus_c lim(q), \quad q_{max} = cen(q) \oplus_c lim(q) \end{aligned} \tag{2.3}$$

where d_{out} represents the distance of the entity to limits of the hyperboloid and d_{in} is the distance of the entity from the hyperboloid's border to its center. γ is a scalar weight (set to 0.5 in our experiments) and $\|x\|_1$ is the $L1$-norm of x.

Translation (t): Each relation $r \in R$ is equipped with a relation embedding $r = Hyperboloid_r \in \mathbb{R}^{2d}$. Given an entity embedding $e \in E$, we model its translation (o_t) and distance from the result entities $v_t \in V_t \subseteq E$ (d_t) as follows:

$$o_t = e \oplus_c r, \quad d_t(v) = d_{hyp}(v_t, o_t) \tag{2.4}$$

This provides us with the translated hyperboloid with a new center and larger limit ($\lim(r) \geq 0$). A sample operation is illustrated in Figure 2.7(a).

Intersection ($\cap$): We model the intersection of a set of hyperboloid embeddings $Q_\cap = \{e_1, e_2, e_3, ..., e_n\}$ as $o_\cap$ and entity distance from the result entities $v_\cap \in V_\cap \subseteq E$ as $d_\cap$ where:

$$o_\cap = (\text{cen}(Q_\cap), \lim(Q_\cap)) \tag{2.5}$$

$$\text{cen}(Q_\cap) = \sum_i a_i \odot_c \text{cen}(e_i); \quad a_i = \frac{\exp(f(e_i))}{\sum_j \exp(f(e_i)} \tag{2.6}$$

$$\lim(Q_\cap) = \min(\{\lim(e_1), ..., \lim(e_n)\}) \odot_c \sigma(DS(\{e_1, ..., e_n\}))$$

$$DS(\{e_1, .., e_n\}) = f\left(\frac{1}{n}\sum_{i=1}^{n} f(e_i)\right) \tag{2.7}$$

$$d_\cap(v_\cap) = d_{hyp}(v_\cap, o_\cap) \tag{2.8}$$

where $\odot_c$ is the Möbius scalar product, $f(.) : \mathbb{R}^{2d} \to \mathbb{R}^d$ is the multilayer perceptron (MLP), $\sigma(.)$ is the sigmoid function and $DS(.)$ is the permutation invariant deep architecture, namely,

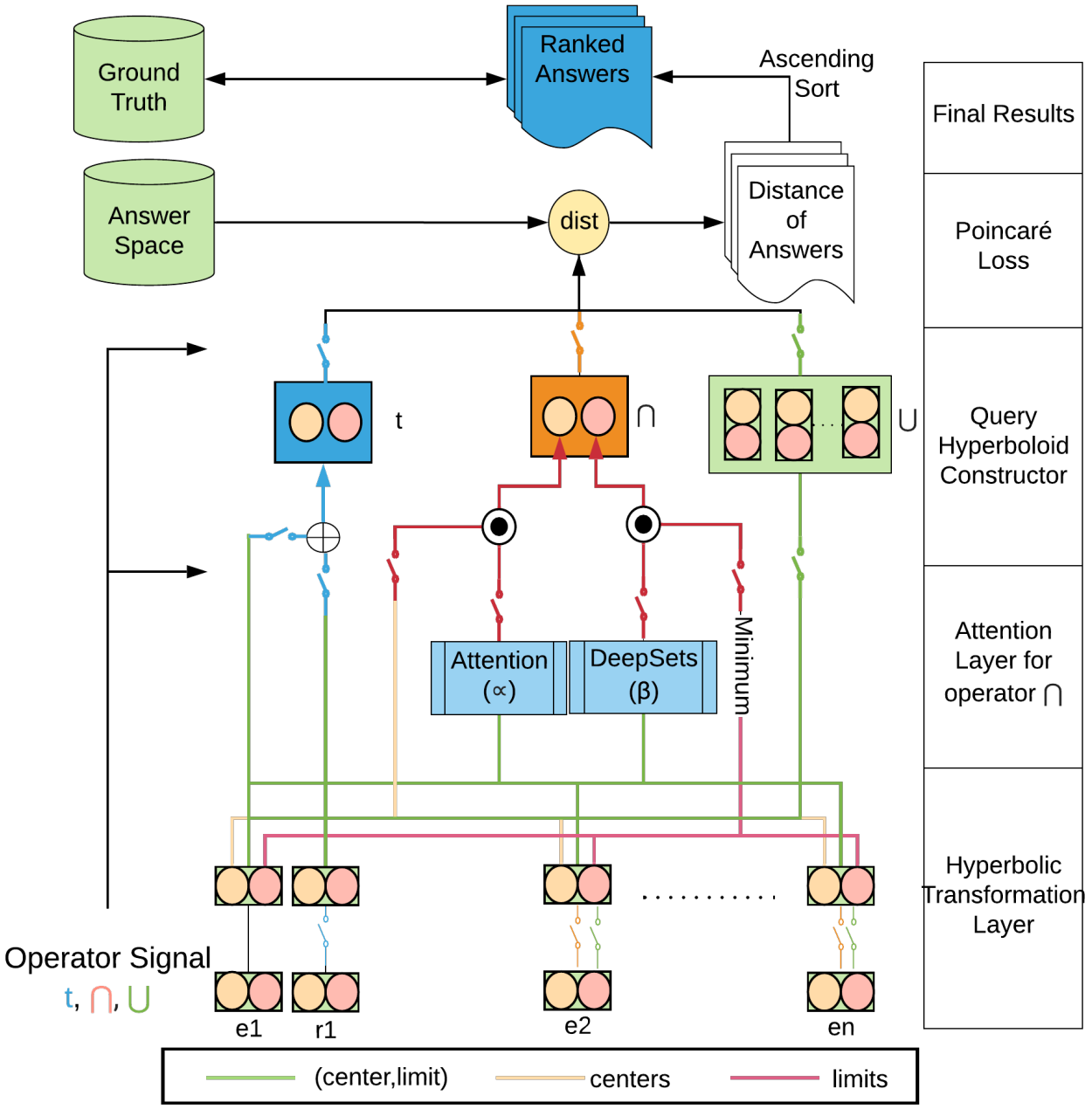


Figure 2.6: An overview of the proposed HypE architecture. The architecture utilizes a switch mechanism to connect/disconnect different layers according to the query operator signal (t,∩,∪). The blue, red and green switches are connected for translation, intersection and union operations, respectively (and disconnected otherwise). The yellow and pink circles depict the center and limit of KG units, respectively. This figure is best viewed in color.

DeepSets [150]. $Min(.)$ and exp(.) are element-wise minimum and exponential functions. The new center and limit are calculated by an attention layer [2] over the hyperboloid centers and DeepSets for shrinking the limits, respectively. Figure 2.7(b) depicts a sample intersection.

Union ($\cup$): Unlike intersection, union operations are not closed under hyperboloids (union of hyperboloids is not a hyperboloid). Hence, the distance of entities from the union query space ($d_\cup$) is defined as the minimum distance from any hyperboloid in the union. For a set of hyperboloid embeddings $Q_\cup = \{e_1, e_2, e_3, ..., e_n\}$, union space is given by $o_\cup$ and distance from result entities $v_\cup \in V_\cup \subseteq E$ by $d_\cup$, where

$$o_\cup = Q_\cup \tag{2.9}$$

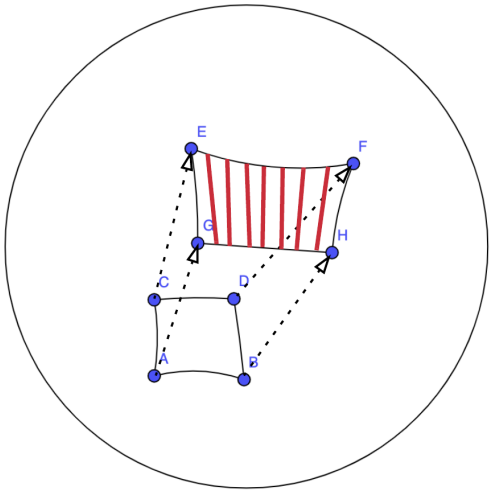
$$d_\cup(v_\cup) = \min\left(\{d_{hyp}(v_\cup, e_i)\ \forall e_i \in o_\cup\}\right) \tag{2.10}$$

Note that, since union is not closed under hyperboloids it cannot be applied before the other operations. We circumvent this problem by utilizing Disjunctive Normal Form (DNF) transformation [104] on our logical queries. This allows us to push all the union operations to the end of our computational graph, thus maintaining validity for all PFOE queries. An outline of HypE's training procedure is given in Algorithm 1.

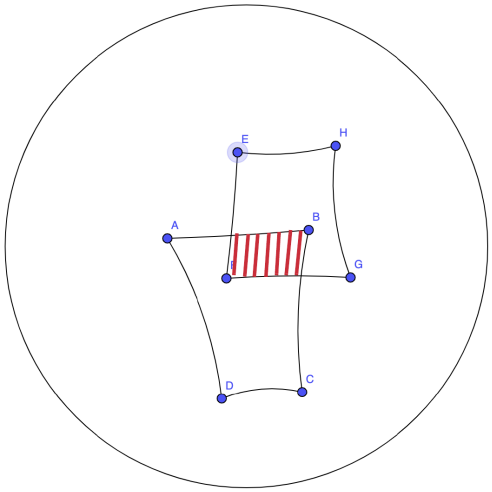
2.3.5 Implementation Details

We implemented HypE in Pytorch [100] on two Nvidia V100 GPUs with 16 GB VRAM. For gradient descent, the model is trained using Reimannian Adam optimizer [4] with an initial learning rate of 0.0001 and standard β values of 0.9 and 0.999. We utilize ReLU [87] as the activation function. Also, we randomly selected 128 negative samples per positive sample in the training phase to learn better discriminative features. For our empirical studies, we learned hyperboloid embeddings of 2×400 dimensions (d=400). Due to the conditionality (i.e., *if* conditions in Algorithm 1) in our computational graph, we employ a switch mechanism between the network layers [40, 80]. The switch mechanism receives an operator signal that defines the operation and accordingly connects/disconnects a layer from the framework. A disconnected switch blocks back-propagation of weight updates to the disconnected layers. This enables a concurrent use of all PFOE queries to update the entity and relation embeddings. For an input query Q and resultant entities V, Algorithm 1 provides the pseudocode of our overall framework to learn representations of entities E and relation R. The algorithm describes the three main operations, namely, translation (lines 4-7), intersection (lines 8-11), and union (lines 12-15) [2].

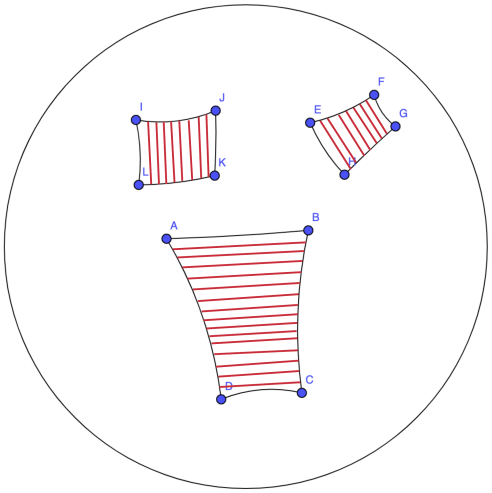
[2]Implementation code: https://github.com/amazon-research/hyperbolic-embeddings



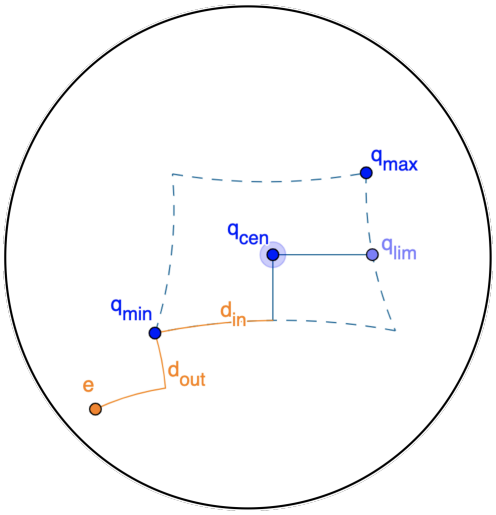
(a) Translation



(b) Intersection



(c) Union



(d) Hyperbolic distance

Figure 2.7: PFOE queries and hyperbolic distance in a Poincaré geodisc.

Algorithm 1: HypE algorithm

Input: Training data $D_t, D_\cap, D_\cup$, which are set of all (query (Q), result (V)) for translation, intersection, and union, respectively;

Output: Entity E and Relation R hyperboloids;

Randomly initialize $e \in E$ and $r \in R$ $(e, r \in \mathbb{H}^{2d})$;
for *number of epochs; until convergence* **do**
  $l = 0$; # Initialize loss
  for $\{(e, r, V_t) \in D_t\}$ **do**
    $o_t = e \oplus_c r$, from Eq. (2.4)
    # Update loss for translation queries
    $l = l + \sum_{v_t \in V_t} d_{hyp}(v_t, o_t)$
  end
  for $\{(Q_\cap, V_\cap) \in D_\cap\}$ **do**
    $o_\cap = (cen(Q_\cap), lim(Q_\cap))$, from Eq. (2.5)
    # Update loss for intersection queries
    $l = l + \sum_{v_\cap \in V_\cap} d_{hyp}(v_\cap, o_\cap)$
  end
  for $\{(Q_\cup, V_\cup) \in D_\cup\}$ **do**
    $o_\cup = Q_\cup$, from Eq. (2.9)
    # Update loss for union queries
    $l = l + \sum_{v_\cup \in V_\cup} Min\left(d_{hyp}(v_\cup, e_i)\ \forall e_i \in o_\cup\right)$
  end
  # Update E and R with backpropagation
  $E \leftarrow E - \Delta_E l$
  $R \leftarrow R - \Delta_R l$
end
return *E,R*

2.4 Experimental Setup

This section describes the experimental setup that analyzes the performance of HypE on various problems. We aim to study the following research questions:

- **RQ1:** For the task of reasoning over KGs, are hyperboloid embeddings better than the baselines at learning hierarchical relations?
- **RQ2:** What is the contribution of individual components in the HypE model?
- **RQ3:** Do the representations capture relevant data features for the downstream task of anomaly detection?
- **RQ4:** Can hyperboloid embeddings leverage auxiliary semantic information from the entities?
- **RQ5:** Why are hyperbolic representations better at capturing hierarchy than Euclidean space?
- **RQ6:** Can we comprehend the latent representational space obtained by the proposed HypE model?

2.4.1 Datasets

We perform our experimental study on the following standard KG and hierarchical graph datasets:

1. **FB15k** [11] contains knowledge base relation triples and textual mentions of Freebase entity pairs. This dataset contains a large number of simple test triples that can be obtained by inverting the training triples.
2. **FB15k-237** [120] is a subset of FB15k where all the simple inversible relations are removed, so the models can learn and focus on more complex relations.
3. **NELL995** [16] is a KG dataset of relation triples constructed from the 995^{th} iteration of the Never-Ending Language Learning (NELL) system.
4. **DBPedia Hierarchical Taxonomy**[3] is a subset extracted from Wikipedia snapshot that provides multi-level hierarchical taxonomy over 342,782 articles (leaf-nodes).
5. **E-commerce Product Network**[4] is a subsampled product taxonomy from an e-commerce platform.

[3] https://www.kaggle.com/danofer/dbpedia-classes
[4] Proprietary dataset

To be consistent with KG terms, for the hierarchical graph datasets (DBPedia and E-commerce) we consider all the intermediate and leaf nodes as entities and the edges between them as relations. Additionally, we consider two variants for encoding edges. First, all edges are considered identical ($|R| = 1$) and second, where all edges are depth-encoded ($|R| = p$), where p is maximum depth of the hierarchy ($p = 3$ for DBPedia and $p = 4$ for E-commerce dataset). For cross-validation and evaluation, we split the graph KG into three parts: KG_{train}, KG_{valid} and KG_{test} in a $75 : 10 : 15$ ratio for our experiments. More details of the datasets are given in Table 2.1.

Table 2.1: Basic statistics of the datasets including the number of unique entities, relations, and edges.

Dataset	# entities	# relations	# edges
FB15k	14,951	2690	273,710
FB15k-237	14,505	474	149,689
NELL995	63,361	400	107,982
DBPedia	34,575	3	240,942
E-commerce	∼118K	4	∼562K

2.4.2 Baselines

We select our baselines based on the following two criteria:

1. The embedding geometries are closed under the intersection and translation operation, e.g., the translation or intersection of arc-aligned hyperboloids results in an arc-aligned hyperboloid.
2. The baseline can be intuitively extended to all PFOE queries over KG. This is necessary to have a fair comparison with HypE that can leverage all PFOE queries.

We adopt the following state-of-the-art baselines based on geometric diversity and our criterion to compare against HypE:

- **Graph Query Embedding (GQE)** [55] embeds entities and relations as a vector embedding in the Euclidean space.
- **Knowledge Graph Attention Network (KGAT)** [134] embeds entities and relations as a vector embedding in the Euclidean space utilizing attention networks over entities and relations with a TransR loss [76] .
- **Hyperbolic Query Embeddings (HQE)** [45] utilizes manifold transformations (refer to Section 2.3.3) to represent entities and relations as a vector embedding in hyperbolic space.

- **Query2Box (Q2B)** [104] embeds entities and relations as axis-aligned hyper-rectangle or box embeddings in Euclidean space.

Some of the other possible baselines [11, 94, 142], solely, focus on the multi-hop (or translation) problem. They could not be naturally extended to other PFOE queries. Additionally, other geometric variants such as circular and Gaussian embeddings [121, 126] are not closed under intersection (intersection of Gaussians is not a Gaussian).

2.4.3 RQ1: Efficacy of the Query-Search space

To analyze the efficacy of the query space obtained from the HypE model, we compare it against the state-of-the-art baselines on the following reasoning query structures:

1. **Single operator queries** include multi-level translation (1t, 2t, and 3t) multi-entity intersection ($2\cap, 3\cap$) and multi-entity union queries ($2\cup$). 1t, 2t, and 3t denote translation with 1, 2 and 3 consecutive relations, respectively. $2\cap$ and $2\cup$ stand for intersection and union over two entities, respectively. $3\cap$ represents intersection over three entities.

2. **Compound queries** contain multiple operators chained in series to get the final result. Our experiments analyze $\cap t$ (intersection-translation), $t\cap$ (translation-intersection) and $\cup t$ (union-translation).

The above queries are illustrated in Figure 2.8.

We extract the ground truth query-entity pairs by traversing the datasets [104]. The models are trained on queries from KG_{train} and validated on KG_{valid}. The final evaluation metrics are calculated on KG_{test}. We utilize Euclidean norm and hyperbolic distance (given in Eq. (2.3)) to measure the distance between query embeddings and its resultant entities in Euclidean and hyperbolic spaces, respectively. The sorted query-entity distances are the ranked results for the given query.

Given a test query q_{test}, let the true ranked result entities be E_{result} and model's ranked output be $E_{output} = \{e_{o1}, e_{o2}, ..., e_{on}\}$. The evaluation metrics used in our work are **Hits@K** and **Mean Reciprocal Rank (MRR)**. The metrics are given by:

$$HITS@K(q_{test}) = \frac{1}{K}\sum_{k=1}^{K} f\left(e_{ok}\right), \qquad f\left(e_{ok}\right) = \begin{cases} 1, \text{if } e_{ok} \in E_{result} \\ 0, \text{ otherwise} \end{cases}$$

$$MRR(q_{test}) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{f\left(e_{oi}\right)}, \qquad f\left(e_{oi}\right) = \begin{cases} i, \text{if } e_{oi} \in E_{result} \\ \infty, \text{otherwise} \end{cases}$$

[6] Due to the sensitive nature of the proprietary dataset, we only report relative results.

Table 2.2: Performance comparison on HITS@3 of HypE (ours) against the baselines to study the efficacy of the Query-Search space. The columns present the different query structures and averages over them. The final row presents the Average Relative Improvement (%) of HypE compared to Query2Box over all datasets. E-Vector and H-Vector are vectors in Euclidean and hyperbolic space, respectively. Best results for each dataset are shown in bold.

Dataset	**Model**	**1t**	**2t**	**3t**	**2∩**	**3∩**	**2∪**	**∩*t***	***t*∩**	**∪*t***	**Avg**
FB15k	GQE (E-Vector)	.636	.345	.248	.515	.624	.376	.151	.310	.273	.386
	KGAT (E-Vector)	.711	.379	.276	.553	.667	.492	.181	.354	.302	.435
	HQE (H-Vector)	.683	.365	.265	.451	.589	.438	.135	.283	.290	.389
	Q2B (Box)	.786	.413	.303	.590	.710	.608	**.211**	.397	.330	.483
	HypE (Hyperboloid)	**.809**	**.486**	**.365**	**.598**	**.728**	**.610**	.206	**.406**	**.410**	**.513**
FB15k -237	GQE (E-Vector)	.404	.214	.147	.262	.390	.164	.087	.162	.155	.221
	KGAT (E-Vector)	.436	.227	.167	.293	.422	.202	.069	.135	.174	.236
	HQE (H-Vector)	.440	.231	.171	.265	.387	.195	.083	.162	.183	.235
	Q2B (Box)	.467	.240	.186	.324	.453	**.239**	.050	.108	.193	.251
	HypE (Hyperboloid)	**.572**	**.366**	**.255**	**.399**	**.527**	.225	**.145**	**.246**	**.282**	**.335**
NELL 995	GQE (E-Vector)	.417	.231	.203	.318	.454	.200	.081	.188	.139	.248
	KGAT (E-Vector)	.486	.249	.218	.331	.467	.285	.107	.200	.151	.277
	HQE (H-Vector)	.477	.250	.219	.270	.413	.267	.091	.153	.166	.256
	Q2B (Box)	.555	.266	.233	.343	.480	.369	.132	.212	.163	.306
	HypE (Hyperboloid)	**.618**	**.359**	**.312**	**.400**	**.563**	**.441**	**.143**	**.227**	**.278**	**.371**
DBPedia $\lvert R\rvert = 1$	GQE (E-Vector)	.673	.006	N.A.	.873	.879	.402	.160	.668	0.00	.406
	KGAT (E-Vector)	.753	.007	N.A.	.937	.940	.526	.192	.762	0.00	.457
	HQE (H-Vector)	.422	.003	N.A.	**1.00**	**1.00**	.138	.109	.182	**.001**	.296
	Q2B (Box)	.832	.007	N.A.	**1.00**	**1.00**	.649	.224	.856	0.00	.508
	HypE (Hyperboloid)	**.897**	**.009**	N.A.	**1.00**	**1.00**	**.708**	**.294**	**.935**	**.001**	**.546**
DBPedia $\lvert R\rvert = p$	GQE (E-Vector)	.730	.565	N.A.	.873	.879	.534	.213	.705	.027	.504
	KGAT (E-Vector)	.816	.621	N.A.	.937	.940	.699	.255	.804	.030	.567
	HQE (H-Vector)	.456	.182	N.A.	**1.00**	**1.00**	.184	.143	.192	**.143**	.367
	Q2B (Box)	.901	.676	N.A.	**1.00**	**1.00**	.863	.297	.903	.033	.630
	HypE (Hyperboloid)	**.970**	**.756**	N.A.	**1.00**	**1.00**	**.940**	**.388**	**.985**	.046	**.676**
Improv. (%) (HypE vs Q2B)		15.0	39.2	67.0	4.1	4.7	13.0	28.5	12.9	7.3	14.4

Table 2.3: Performance comparison on Mean Reciprocal Rank (MRR) of HypE (ours) against the baselines to study the efficacy of the Query-Search space. The columns present the different query structures and averages over them. The final row presents the Average Relative Improvement (%) of HypE compared to Query2Box over all datasets. E-Vector and H-Vector are vectors in Euclidean and hyperbolic space, respectively. Best results for each dataset are shown in bold.

Dataset	**Model**	**1t**	**2t**	**3t**	**2∩**	**3∩**	**2∪**	**∩*t***	***t*∩**	**∪*t***	**Avg**
FB15k	GQE (E-Vector)	.505	.320	.218	.439	.536	.300	.139	.272	.244	.330
	KGAT (E-Vector)	.565	.352	.243	.471	.573	.393	.167	.311	.270	.372
	HQE (H-Vector)	.543	.339	.233	.384	.506	.350	.125	.249	.259	.332
	Q2B (Box)	.654	.373	.274	.488	.602	.468	**.194**	.339	.301	.410
	HypE (Hyperboloid)	**.673**	**.439**	**.330**	**.495**	**.617**	**.470**	.189	**.347**	**.374**	**.437**
FB15k -237	GQE (E-Vector)	.346	.193	.145	.250	.355	.145	.086	.156	.151	.203
	KGAT (E-Vector)	.373	.205	.165	.280	.384	.179	.068	.130	.170	.217
	HQE (H-Vector)	.376	.209	.169	.253	.352	.173	.082	.156	.179	.217
	Q2B (Box)	.400	.225	.173	.275	.378	**.198**	.105	.180	.178	.235
	HypE (Hyperboloid)	**.490**	**.343**	**.237**	**.339**	**.440**	.186	**.305**	**.410**	**.260**	**.334**
NELL 995	GQE (E-Vector)	.311	.193	.175	.273	.399	.159	.078	.168	.130	.210
	KGAT (E-Vector)	.362	.208	.188	.284	.410	.227	.103	.179	.141	.234
	HQE (H-Vector)	.355	.209	.189	.232	.363	.213	.088	.137	.155	.216
	Q2B (Box)	.413	.227	.208	.288	.414	.266	.125	.193	.155	.254
	HypE (Hyperboloid)	**.460**	**.306**	**.279**	**.336**	**.486**	**.318**	**.135**	**.207**	**.264**	**.310**
DBPedia $\lvert R\rvert = 1$	GQE (E-Vector)	.502	.005	N.A.	.749	.773	.32	.154	.597	0.00	.344
	KGAT (E-Vector)	.561	.006	N.A.	.804	.825	.419	.185	.682	0.00	.387
	HQE (H-Vector)	.314	.003	N.A.	**.859**	**.879**	.110	.105	.163	**.001**	.270
	Q2B (Box)	.619	.006	N.A.	.840	.863	.468	.212	.779	0.00	.421
	HypE (Hyperboloid	**.668**	**.008**	N.A.	.840	.863	**.511**	**.278**	**.853**	**.001**	**.447**
DBPedia $\lvert R\rvert = p$	GQE (E-Vector)	.544	.421	N.A.	.651	.656	.398	.159	.526	.020	.375
	KGAT (E-Vector)	.608	.463	N.A.	.698	.700	.521	.190	.599	.022	.422
	HQE (H-Vector)	.339	.135	N.A.	**.744**	**.744**	.137	.106	.143	**.106**	.273
	Q2B (Box)	.670	.503	N.A.	**.744**	**.744**	.642	.221	.672	.025	.469
	HypE (Hyperboloid)	**.722**	**.563**	N.A.	**.744**	**.744**	**.700**	**.289**	**.733**	.034	**.503**
Improv. (%) (HypE vs Q2B)		14.9	38.4	59.8	4.4	5.0	12.9	39.5	17.9	64.9	18.5

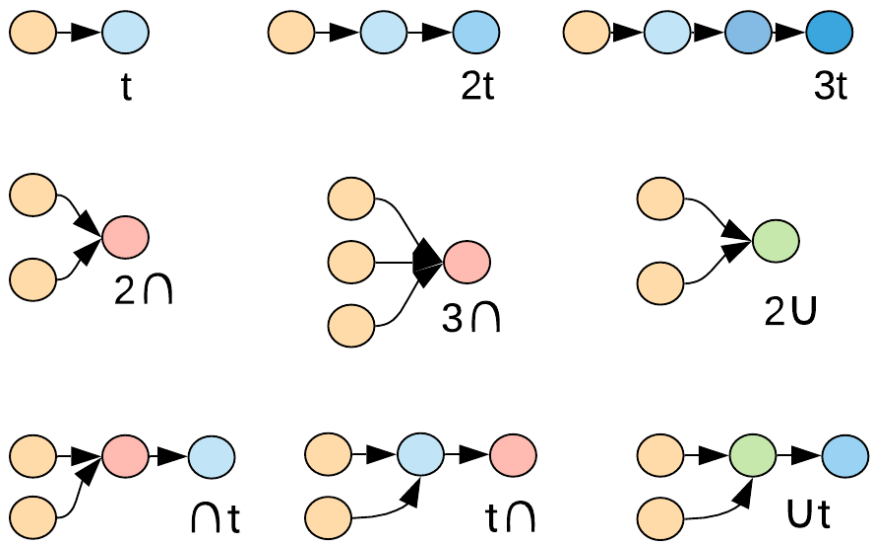


Figure 2.8: Logical query structures designed to compare HypE against baselines. The blue, red, and green units denote translation, intersection, and union operations, respectively.

Table 2.4: Performance comparison of MRR of HypE (ours) against the baselines on an e-commerce dataset for logical queries. We assume GQE (E-vector) as a baseline and report relative improvements against that for all the methods. The numbers are in percentages. Best results for each dataset are shown in bold.[5]

Metrics		**Hits@3**								
Dataset	**Model**	**1t**	**2t**	**3t**	**2∩**	**3∩**	**2∪**	**∩*t***	***t*∩**	**∪*t***
$\|R\| = 1$	GQE (E-Vector)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	KGAT (E-Vector)	12.12	10.9	0.0	0.0	-36.2	-57.3	**53.8**	**2850**	30.8
	HQE (H-Vector)	-34.8	-51.6	0.0	-34.7	-56.6	0.0	0.0	0.0	-57.7
	Q2B (Box)	30.3	4.7	**200**	23.5	7.5	0.0	-1.9	-50	32.7
	HypE (Hyperboloid)	**38.6**	**98.4**	**200**	**38.8**	**79.2**	**100**	-1.9	-50	**119.2**
$\|R\| = p$	GQE (E-Vector)	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	**0.0**	0.0
	KGAT (E-Vector)	12.2	10.7	0.0	11.5	9.5	0.0	**0.0**	**0.0**	30.2
	HQE (H-Vector)	38.8	92.9	350	38.5	90.5	600	**0.0**	**0.0**	0.0
	Q2B (Box)	**0.0**	200	0.0	**11.8**	**0.0**	59.4	-75	**100**	0.0
	HypE (Hyperboloid)	**122.3**	**298.2**	**8350**	**192.3**	**471.4**	**9900**	**0.0**	**0.0**	**416.3**

Table 2.5: Performance comparison of Hits@3 of HypE (ours) against the baselines on an e-commerce dataset for different logical queries. We assume GQE (E-vector) as a baseline and report relative improvements against that for all the methods. The numbers are in percentages. Best results for each dataset are shown in bold.[6]

Metrics		**Mean Reciprocal Rank**								
Dataset	**Model**	**1t**	**2t**	**3t**	**2∩**	**3∩**	**2∪**	**∩*t***	***t*∩**	**∪*t***
$\|R\| = 1$	GQE (E-Vector)	0.0	0.0	0.0	**0.0**	**0.0**	0.0	0.0	0.0	0.0
	KGAT (E-Vector)	200	**300**	0.0	-36.2	-57.3	24.4	-60	-45.5	0.0
	HQE (H-Vector)	-34.8	-51.6	0.0	-34.7	-56.6	0.0	0.0	0.0	-57.7
	Q2B (Box)	0.0	50	0.0	-59.6	-99	53.7	-80	-81.8	0.0
	HypE (Hyperboloid)	0.0	**50**	**1500**	-59.6	-99	**107.3**	-80	-81.8	**1100**
$\|R\| = p$	GQE (E-Vector)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	KGAT (E-Vector)	**0.0**	**2600**	0.0	-8.8	**0.0**	31.3	-25	0.0	0.0
	HQE (H-Vector)	-50	0.0	**19400**	**11.8**	**0.0**	0.0	-75	0.0	**14400**
	Q2B (Box)	**0.0**	200	0.0	**11.8**	**0.0**	59.4	-75	**100**	0.0
	HypE (Hyperboloid)	**0.0**	**2600**	**19400**	**11.8**	**0.0**	**415.6**	-75	**100**	**14400**

From the results in Table 2.2 and 2.3, we observe that, on average, HypE outperforms the current baselines in translation, single, and compound operator queries by 15%-67%, 4%-13%, and 7%-28%, respectively. The performance improvement linearly increases with higher query depth ($1t < 2t < 3t$). Furthermore, we notice that unique depth encoding ($|R| = d$) outperforms identical depth encoding ($|R| = 1$) by 23% in DBPedia. Because of our subsampling strategy, the E-commerce Product Network is disjoint (i.e., there are several intermediate nodes that do not share children). Hence, the number of intersection queries are extremely low and insufficient for training HypE or baselines. Hence, there are extremely high values in the results as shown in Table 2.4 and Table 2.5, where we report relative performance improvements with respect to the GQE baseline.

2.4.4 RQ2: Ablation Study

In this section, we empirically analyze the importance of different layers adopted in the HypE model. For this, we experiment with different variations of the center aggregation layer; Average (HypE-Avg), Attention (HypE) (refer to Eq. (2.6)) and Deepsets (HypE-DS) (refer to Eq. (2.7)). Furthermore, we test the exclusion of intersection and unions to comprehend their importance in the representation learning process. We adopt two variants of HypE; one trained on only 1t queries (HypE-Avg-1t) and the other trained on all translation queries (HypE-Avg-1,2,3t). Table 2.6 and 2.7 presents the performance metrics of different variants on the query processing task.

Firstly, we observe that the exclusion of intersection and union queries results in a significant

Table 2.6: Ablation study results on the metric of Hits@3. The first column presents the model variants compared against the final HypE model. Avg-1t and Avg-1,2,3t variants only utilize average center aggregation because other aggregation variants only apply when intersections are involved. HypE-TC presents the HypE variant with trainable curvature. The metrics reported in the table are averaged across evaluation on all the datasets. Best results are shown in bold.

Dataset	**Model**	**1t**	**2t**	**3t**	**2∩**	**3∩**	**2∪**	**∩*t***	***t*∩**	**∪*t***	**Avg**
FB15k	HypE-Avg-1t	.803	.304	.202	.468	.534	.606	.115	.256	.228	.395
	HypE-Avg-1,2,3t	.803	.392	.282	.527	.612	.608	.155	.305	.317	.446
	HypE-Avg	.802	.479	.361	.585	.690	.609	.194	.353	.405	.497
	HypE-DS	.777	.479	.354	.596	.717	.564	.192	.387	.403	.496
	HypE-TC	**.809**	**.486**	**.365**	**.598**	**.729**	**.610**	**.206**	**.406**	**.410**	**.513**
	HypE (final)	**.809**	**.486**	**.365**	**.598**	.728	**.610**	**.206**	**.406**	**.410**	**.513**
FB15k -237	HypE-Avg-1t	.567	.229	.141	.312	.387	.223	.081	.155	.157	.258
	HypE-Avg-1,2,3t	.567	.295	.197	.351	.444	.224	.109	.185	.218	.292
	HypE-Avg	.567	.361	.252	.390	.500	**.225**	.137	.214	.279	.325
	HypE-DS	.549	.361	.247	.398	.519	.208	.135	.234	.277	.324
	HypE-TC	**.573**	**.366**	**.256**	**.399**	**.528**	**.225**	**.146**	**.246**	**.283**	**.335**
	HypE (final)	.572	**.366**	.255	**.399**	.527	**.225**	.145	**.246**	.282	**.335**
NELL 995	HypE-Avg-1t	.614	.225	.173	.313	.413	.438	.080	.143	.155	.286
	HypE-Avg-1,2,3t	.613	.290	.241	.352	.474	.439	.108	.170	.215	.323
	HypE-Avg	.612	.354	.309	.391	.534	.440	.135	.197	.275	.359
	HypE-DS	.594	.354	.303	.399	.554	.408	.134	.216	.273	.359
	HypE-TC	**.618**	**.360**	**.313**	**.400**	**.563**	**.442**	**.144**	**.228**	**.278**	**.371**
	HypE (final)	**.618**	.359	.312	**.400**	**.563**	.441	.143	.227	**.278**	**.371**
DBPedia $\lvert R\rvert = p$	HypE-Avg-1t	.963	.473	N.A.	.782	.734	.933	.216	.621	.026	.521
	HypE-Avg-1,2,3t	.962	.609	N,A.	.880	.841	.936	.291	.739	.036	.588
	HypE-Avg	.961	.745	N.A.	.978	.948	.938	.366	.856	.045	.655
	HypE-DS	.932	.745	N.A.	.997	.985	.869	.362	.938	.045	.654
	HypE-TC	**.971**	**.757**	N.A.	**1.00**	**1.00**	**.941**	**.388**	**.986**	**.047**	**.677**
	HypE (final)	.970	.756	N.A.	**1.00**	**1.00**	.940	**.388**	.985	.046	.676
E-commerce $\lvert R\rvert = 1$	HypE-Avg-1t	0.0	0.0	0.0	0.0	**0.0**	0.0	**0.0**	0.0	0.0	0.0
	HypE-Avg-1,2,3t	0.0	28.6	39.4	12.5	**0.0**	0.5	**0.0**	**50.0**	39.8	13.0
	HypE-Avg	-0.3	57.1	77.7	25.0	**0.0**	0.9	**0.0**	**50.0**	78.7	26.0
	HypE-DS	-3.3	57.1	74.5	**27.5**	**0.0**	-6.8	**0.0**	**50.0**	76.9	26.0
	HypE-TC	**0.7**	**60.0**	**80.9**	**27.5**	**0.0**	**0.9**	**0.0**	**50.0**	**81.5**	**31.0**
	HypE (final)	**0.7**	59.3	79.8	**27.5**	**0.0**	**0.9**	**0.0**	**50.0**	80.6	30.0
E-commerce $\lvert R\rvert = p$	HypE-Avg-1t	**0.0**	0.0	0.0	0.0	**0.0**	0.0	**0.0**	0.0	0.0	0.0
	HypE-Avg-1,2,3t	-1.0	28.5	36.8	13.3	**0.0**	0.0	**0.0**	100	35.3	12.6
	HypE-Avg	-2.3	57.0	72.8	23.3	**0.0**	0.0	**0.0**	100	**70.6**	24.2
	HypE-DS	-4.9	58.3	71.9	**30.0**	**0.0**	-6.1	**0.0**	100	69.4	26.3
	HypE-TC	-1.3	**59.6**	**75.4**	**30.0**	**0.0**	**0.6**	**0.0**	**200**	**70.6**	**28.4**
	HypE (final)	-1.3	58.9	**75.4**	26.7	**0.0**	**0.6**	**0.0**	100	**70.6**	**28.4**

Table 2.7: Ablation study results on the metric of Mean Reciprocal Rank (MRR). The first column presents the model variants compared against the final HypE model. Avg-1t and Avg-1,2,3t variants only utilize average center aggregation because other aggregation variants only apply when intersections are involved. HypE-TC presents the HypE variant with trainable curvature. The metrics reported in the table are averaged across evaluation on all the datasets. Best results are shown in bold.

Dataset	**Model**	**1t**	**2t**	**3t**	**2∩**	**3∩**	**2∪**	**∩*t***	***t*∩**	**∪*t***	**Avg**
FB15k	HypE-Avg-1t	**.683**	.276	.188	.392	.448	.467	.111	.221	.219	.338
	HypE-Avg-1,2,3t	.675	.355	.257	.438	.513	.467	.144	.262	.296	.380
	HypE-Avg	.667	.433	.325	.483	.578	.467	.177	.302	.373	.422
	HypE-DS	.649	.437	.324	**.506**	**.620**	.439	.176	.333	.370	.428
	HypE-TC	.674	**.439**	**.331**	.495	.617	**.470**	**.189**	**.348**	**.375**	**.437**
	HypE (final)	.673	**.439**	.330	.495	.617	**.470**	**.189**	.347	.374	**.437**
FB15k	HypE-Avg-1t	**.498**	.216	.135	.269	.319	.185	.179	.261	.152	.258
-237	HypE-Avg-1,2,3t	.492	.277	.185	.300	.366	.185	.233	.309	.206	.291
	HypE-Avg	.486	.338	.234	.331	.412	.185	.286	.357	.259	.323
	HypE-DS	.473	.341	.233	**.347**	**.442**	.174	.285	.393	.257	.327
	HypE-TC	.490	**.344**	**.238**	.339	.440	**.186**	**.306**	**.411**	**.260**	**.335**
	HypE (final)	.490	.343	.237	.339	.440	**.186**	.305	.410	**.260**	.334
NELL	HypE-Avg-1t	**.467**	.192	.159	.266	.353	.316	.079	.131	.154	.240
995	HypE-Avg-1,2,3t	.462	.247	.217	.297	.404	.316	.103	.156	.209	.270
	HypE-Avg	.456	.302	.275	.328	.455	.316	.127	.180	.263	.299
	HypE-DS	.444	.304	.274	**.344**	**.488**	.297	.126	.198	.261	.304
	HypE-TC	.461	**.306**	**.279**	.336	.487	**.319**	**.135**	**.207**	**.264**	**.310**
	HypE (final)	.460	**.306**	**.279**	.336	.486	.318	**.135**	**.207**	**.264**	**.310**
DBPedia	HypE-Avg-1t	.713	.374	N.A.	.589	.540	.696	.170	.467	.020	.390
$\|R\| = p$	HypE-Avg-1,2,3t	.724	.455	N.A.	.658	.619	.696	.221	.553	.027	.438
	HypE-Avg	.715	.555	N.A.	.726	.697	.696	.271	.638	.034	.486
	HypE-DS	.697	.560	N.A.	.741	.738	.654	.270	.703	.034	.490
	HypE-TC	**.722**	**.563**	N.A.	**.745**	**.745**	**.701**	**.289**	**.733**	**.034**	**.504**
	HypE (final)	**.722**	**.563**	N.A.	.744	.744	.700	**.289**	**.733**	**.034**	.503

performance decrease by 25% (Avg-1,2,3t vs HypE). Furthermore, removing deeper queries such as 2t and 3t, also results in an additional decrease by 17% (Avg-1t vs Ag-1,2,3t). The tests on different aggregation layers prove that Attention is better than average and Deepsets by 23.5% and 14.5%, respectively. Additionally, we notice that employing a trainable curvature results in a slight performance improvement of 0.3%. However, given the incremental performance boost but significant increase in the number of parameters (∼10K) that the trainable curvature adds to the framework, we ignore this component in the final HypE model.

As explained in Section 2.3.4, the final HypE model adopts a Poincaré ball manifold with non-trainable curvature, in addition to attention and Deepsets layer for center and limit aggregation, respectively. Additionally, HypE leverages all PFOE queries.

2.4.5 RQ3: Performance on Anomaly Detection

In this experiment, we utilize the entity and relation representations, trained on the DBPedia Hierarchical Taxonomy and E-commerce Product Network with query processing task, to identify products that might be anomalously categorized. We consider identifying the anomalous children by three levels of parents (i.e., taxonomy levels); P_1, P_2 and P_3. The motivating application is to categorize items that are potentially mis-categorized by sellers into the more relevant (correct) part of the product taxonomy.

Table 2.8: Results on Miscategorized Article Anomaly Detection in DBPedia dataset. Best results are shown in bold and the second best results are underlined. P, R, and F1 represent Precision, Recall, and F-score, respectively.

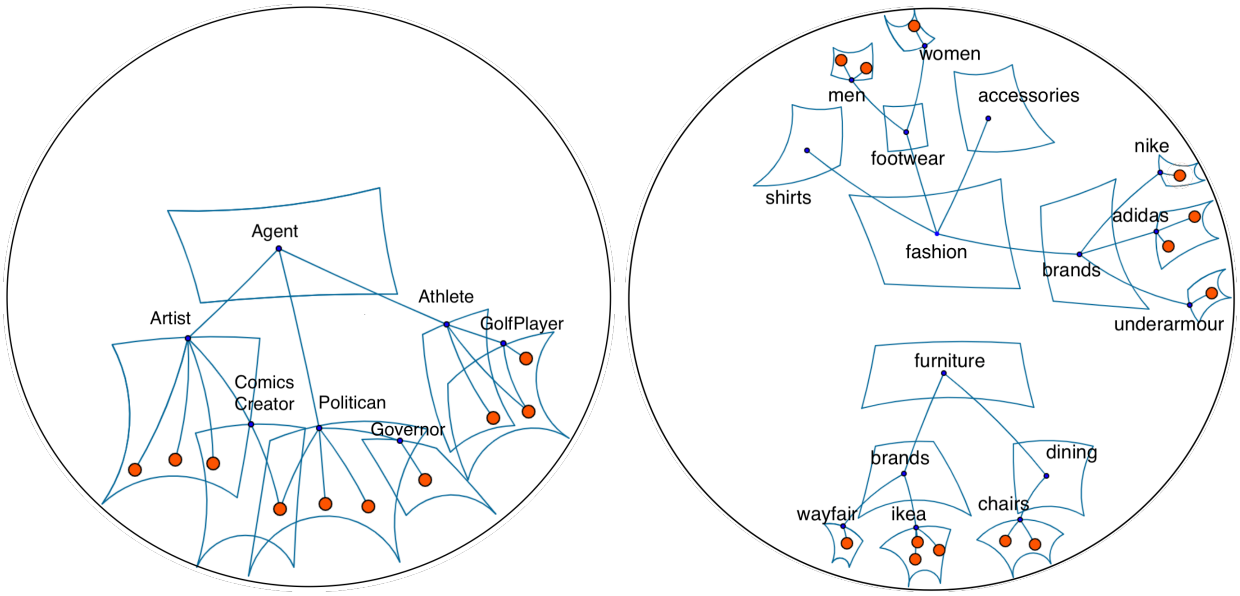
	P-Level	**1**			**2**			**3**		
Dataset	**Models**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
DBPedia	GQE	.512	.369	.428	.549	.446	.492	.576	.409	.479
$\|R\| = p$	KGAT	.523	.375	.437	.552	.448	.495	.578	.416	.484
	HQE	.529	.385	.446	.556	.45	.497	.586	.424	.492
	Q2B	.589	<u>.479</u>	.527	.589	.479	.528	.597	.481	.532
	HypE	.590	<u>.479</u>	.528	<u>.648</u>	.482	.552	.650	<u>.486</u>	<u>.557</u>
	HypE-SI	<u>.591</u>	<u>.479</u>	<u>.529</u>	<u>.648</u>	<u>.483</u>	<u>.553</u>	<u>.651</u>	<u>.486</u>	<u>.557</u>
	HypE-SC	**.601**	**.501**	**.546**	**.662**	**.563**	**.608**	**.705**	**.563**	**.626**

We construct a pseudo-tree, where all parent nodes are infused with 10% noise of randomly sampled anomalous leaf nodes from different parts of the dataset. The goal of the model is to learn representations from this pseudo-tree and identify anomalous leaf nodes of the immediate parent nodes. From the set of all intermediate nodes KG_P, given a parent $p \in KG_P$ and its original set of children $C_+ = children(p)$ and randomly sampled set of

[7]Due to the sensitive nature of the proprietary dataset, we only report relative results.

Table 2.9: Results on Miscategorized Product Anomaly Detection in E-commerce Product Networks. Best results are shown in bold and the second best results are underlined. The improvements are relative to the GQE baseline. P, R, and F1 represent Precision, Recall, and F-score, respectively.[7]

	P-Level	**1**			**2**			**3**		
Dataset	**Models**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
E-commerce $\|R\| = p$	GQE	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	KGAT	2.2	1.8	2.1	0.6	0.5	0.7	0.2	1.6	1.1
	HQE	3.4	4.5	4.1	1.2	1.0	1.1	1.7	3.5	2.8
	Q2B	15.1	29.9	23.4	7.2	7.4	7.4	3.6	17.5	11.3
	HypE	15.3	29.9	23.4	18.0	8.4	12.8	16.6	18.5	17.2
	HypE-SI	15.5	29.9	23.7	18.0	8.4	12.5	16.4	18.8	17.2
	HypE-SC	**17.4**	**35.8**	**27.5**	**20.6**	**26.4**	**23.7**	**22.3**	**37.6**	**30.8**



(a) DBPedia Taxonomy (b) E-commerce Product Networks

Figure 2.9: Visualization of HypE representations for samples from hierarchical datasets in Poincaré ball. The hyperboloids have been scaled up 10 times for better comprehension. The blue (intermediate nodes) circles are annotated with their entity names and orange circles (leaf nodes) depict articles and products in (a) and (b), respectively.

Table 2.10: Example of Anomalies in the E-commerce dataset. The models predict "MISCAT" and "TRUE" tags for mis-categorized and truly-categorized items, respectively. Correct and Incorrect tags are given in green and red color, respectively. HypE performs better than Query2Box (Q2B) as we consider higher level of parents because hyperbolic space is better at capturing hierarchical features. Also, HypE-SC is able to utilize semantic information to improve prediction.

Product Title	**Parent**		**Prediction**		
	Category	P-level	Q2B	HypE	HypE-SC
ASICS Women's Gel-Venture 7 Trail Running Shoes, 5.5M, Graphite Grey/Dried Berry	Rompers	1	TRUE	TRUE	MISCAT
inktastic Family Cruise Youth T-Shirt Youth X-Large (18-20) Pacific Blue 35074	Calvin Klein	1	TRUE	MISCAT	MISCAT
Calvin Klein Men's Cotton Classics Multipack Crew Neck T-Shirts	Calvin Klein	1	MISCAT	MISCAT	TRUE
Epic Threads Big Girls Paint Splatter Distressed Girlfriend Denim Jeans (Dark Wash, 10)	Wool & Blends	2	MISCAT	MISCAT	TRUE
New Balance Women's Crag V1 Fresh Foam Trail Running Shoe, Black/Magnet/Raincloud	Wool & Blends	2	TRUE	MISCAT	MISCAT
Fifth Harmony Vintage Photo Blue T Shirt (M)	Customer Segment	2	TRUE	TRUE	MISCAT
Billy Bills Playoff Shirt Buffalo T-Shirt Let's Go Buffalo Tee	Customer Segment	2	MISCAT	TRUE	TRUE
Kanu Surf Toddler Karlie Flounce Beach Sport 2-Piece Swimsuit, Ariel Blue, 4T	Brand Stores	3	TRUE	MISCAT	MISCAT
The North Face Infant Glacier ¼ Snap, Mr. Pink, 0-3 Months US Infant	Brand Stores	3	MISCAT	TRUE	TRUE
Artisan Outfitters Mens Surfboard Shortboard Batik Cotton Hawaiian Shirt	Specialty Stores	3	MISCAT	TRUE	TRUE
PUMA Unisex-Kid's Astro Kick Sneaker, Peacoat-White-teamgold, 3 M US Little Kid	Specialty Stores	3	TRUE	TRUE	MISCAT

anomalous children $C_- = children\,(KG_P \setminus \{p\})$, the aim here is to identify $c \in C_-$ from $C_+ \cup C_-$. We use Precision, Recall and F1-score as the evaluation metrics for this experiment.

The results (given in Table 2.8 and Table 2.9) show that, although HypE has comparable performance to baselines at P_1, it outperforms the baselines by more than 5% at P_2 and P_3. This demonstrates the robustness of HypE to noisy data and its capability of capturing relevant hierarchical features (as F1 on $P_3 > P_2 > P_1$) for downstream tasks. Furthermore, the specific task is critical in e-commerce search as irrelevant results impede a smooth user experience. Table 2.10 presents some qualitative examples from the E-commerce dataset.

2.4.6 RQ4: Leveraging Semantic Information

KGs generally also contain additional auxiliary information within the entities. In this section, we test the possibility of leveraging the semantic information in the DBPedia (article titles) and E-commerce (product titles) dataset to improve representations. We study two methods to connect HypE with FastText embeddings [8] of the corresponding titles:

- **Semantic Initiation (SI)** initiates the HypE's entities with semantic embeddings and learns new HypE-SI embeddings with the query-processing task (given in Section 2.4.3).

- **Semantic Collaboration (SC)** concatenates the HypE's pre-trained entity representations with semantic embeddings.

We investigate the performance of these methods on the task of anomaly detection. The results of the experiments are given in Tables 2.8 and 2.9. The results demonstrate that HypE-SI shows no significant performance improvement over HypE. That is, a good semantic initialization of the vectors does not result in better representations. This is reasonable, since the semantic embeddings are learnt in the Euclidean space, and several transformations occur between the initialization and final representations. This also means that the learning framework is robust to initialization. We observe a performance improvement of $2\% - 8\%$ in case of HypE-SC when compared to HypE. This suggests the ubiquity of HypE since hierarchical representations can be independently augmented with other auxiliary features to solve more complex tasks. From the examples given in Table 2.10, we can observe that HypE-SI is able to leverage semantic information from *product title* and *category name* to enrich HypE's hierarchical information to produce better predictions. The additional semantic information is especially useful for product miscategorization. In the absence of semantic information, HypE will merely learn representations based on the noisy graph and will lose discriminative information between outliers and correct nodes.

2.4.7 RQ5: Hyperbolic vs Euclidean distances

To better analyze the impact of adopting hyperbolic space, we need to understand its distinction from the Euclidean space in handling hierarchy. For this, we study the intra-level and inter-level Euclidean and hyperbolic distance between entities at different levels of the dataset. Let us say E_p is the set of entities at level P_p in the E-commerce Product Networks dataset. For the analysis, we calculate two sets of distances; intra-level (Δ_{intra}) and inter-level (Δ_{inter}) distance as follows:

$$\Delta_{intra}(P_p) = \frac{\sum_{i=0}^{|E_p|} \sum_{j=i}^{|E_p|} \delta(e_i, e_j)}{|E_p|\,(|E_p| - 1)} \tag{2.11}$$

$$\Delta_{inter}(P_{p1}, P_{p2}) = \frac{\sum_{i=0}^{|E_{p1}|} \sum_{j=0}^{|E_{p2}|} \delta(e_i, e_j)}{|E_{p1}| \times |E_{p2}|} \tag{2.12}$$

$\delta(.\ ,\ .)$ is replaced with Euclidean norm (on Query2Box representations) and hyperbolic distance (d_{hyp} on HypE representations) to understand the difference between the hierarchical separation of entities in the two spaces.

In the Δ_{intra} results (depicted in Figure 2.10), we observe that, with increasing level of hierarchy the distance between entities at different levels remains constant in the case of Euclidean space and shows a clear decreasing trend for hyperbolic space. This indicates denser clustering of entities at the same level. Additionally, the Δ_{inter} results, illustrated in Figure 2.11, depict a linear increase in distance between inter-level entities in the Euclidean space and a superlinear growth in the hyperbolic space. This shows that hyperbolic space also learns clusters such that inter-level entities are farther apart compared to Euclidean space. This nature of inter-level discrimination and intra-level aggregation demonstrates the superior ability of hyperbolic spaces at capturing hierarchical features.

2.4.8 RQ6: Visualization of the Poincaré ball

We extract representative examples of different operations from our dataset and visualize them in a 2-dimensional Poincaré ball of unit radius. We employ Principal Component Analysis (PCA) [6] for dimensionality reduction ($\mathbb{R}^{2d} \rightarrow \mathbb{R}^2$) of the hyperboloids in Poincaré ball.

Figure 2.9 depicts the HypE representations in a Poincaré ball manifold. Notice that the density of nodes increases superlinearly from the center towards the circumference, which is analogous to the superlinear increase in the number of nodes from root to the leaves. Thus, HypE is able to learn a better distinction between different hierarchy levels, when compared to the Euclidean distance-based baselines, which conform to a linear increase. Furthermore, we observe that hyperboloid intersections in DBPedia Taxonomy (Figure 2.9(a)) capture entities with common parents. Also, the disjoint nature of E-commerce Product Networks

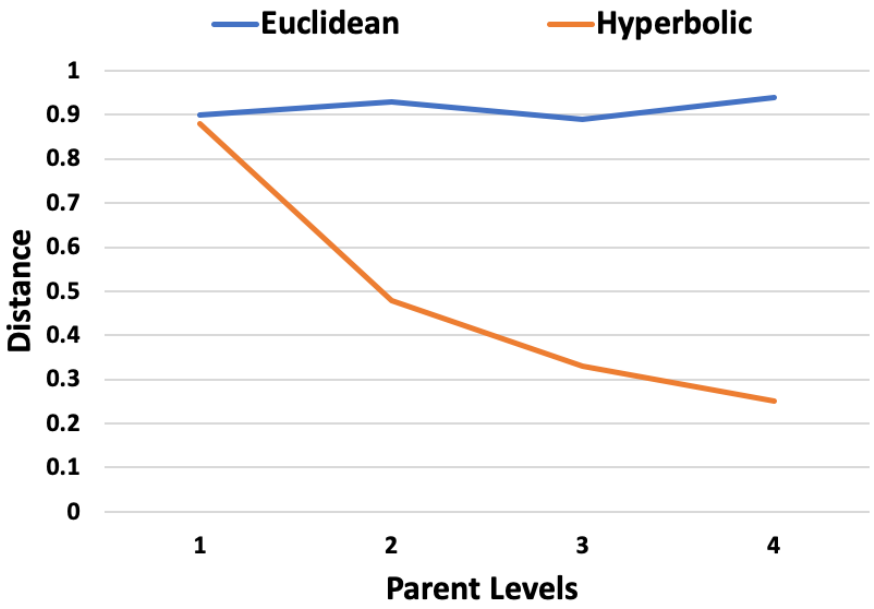


Figure 2.10: Intra-level Euclidean (Q2B) and Hyperbolic (HypE) distances. The graph presents Δ_{intra} of entity sets at different hierarchy levels, given on the x-axis.

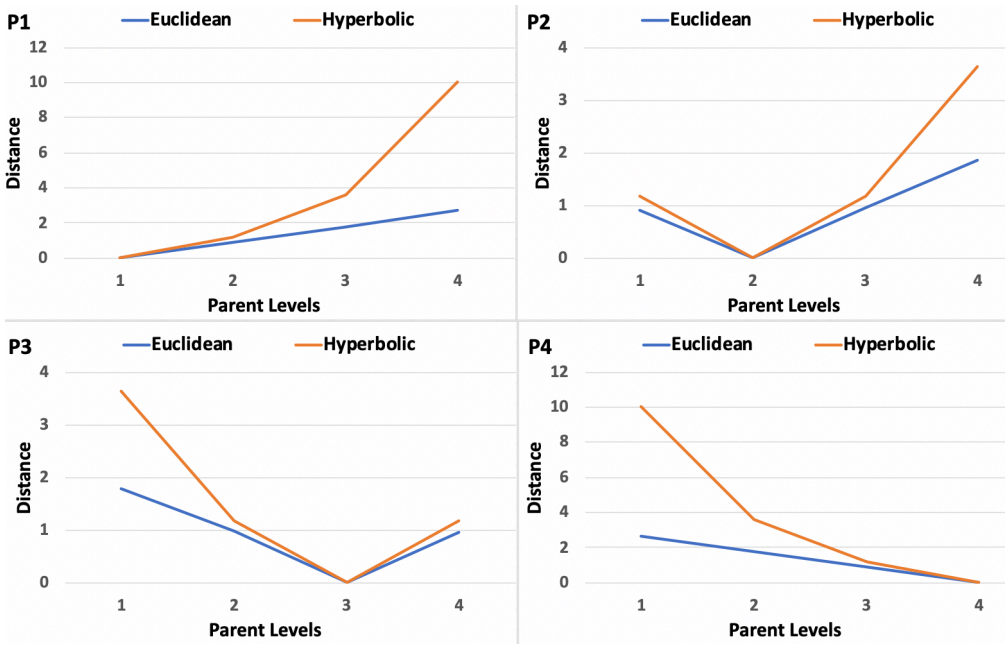


Figure 2.11: Inter-level Euclidean (Q2B) and Hyperbolic (HypE) distances. Each graph presents Δ_{inter} between entity pairs of a source hierarchical level, given by the graph label and the other hierarchy levels in the dataset, given on the x-axis.

(Figure 2.9(b)) is illustrated by disjoint non-intersecting hyperboloids in the latent space. In addition, we can also notice that the learnable limit parameter adjusts the size of hyperboloids to accommodate its varying number of leaf nodes. Thus, the complex geometry of HypE is able to improve its precision over vector baselines that, generally, utilize static thresholds over distance of the resultant entities from query points.

2.5 Summary

In this chapter, we presented Hyperboloid Embeddings (HypE) model, a novel self-supervised learning framework that utilizes dynamic query-reasoning over KGs as a proxy task to learn representations of entities and relations in a hyperbolic space. We demonstrate the efficacy of a hyperbolic query-search space against state-of-the-art baselines over different datasets. Furthermore, we also show the effectiveness of hyperboloid representations in complex downstream tasks and study methods that can leverage node's auxiliary information to enrich HypE features. Additionally, we analyze the contribution of HypE's individual components through an ablation study. Finally, we present our hyperboloid representations in a 2-dimensional Poincaré ball for better comprehensibility.

Chapter 3

ANTHEM: Attentive Hyperbolic Entity Model for Product Search

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 chapter, 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.

3.1 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 [39, 85, 115].

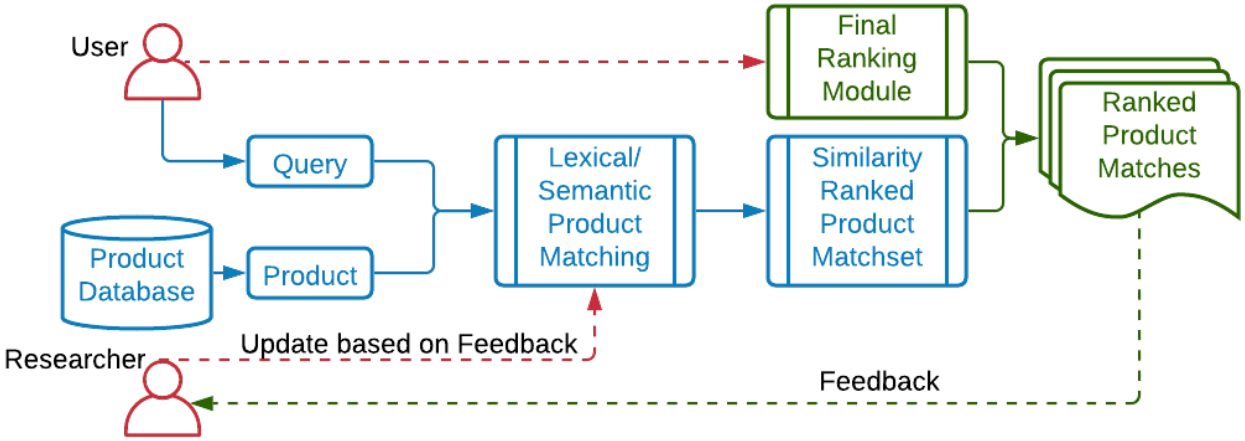


Figure 3.1: Product search framework. This chapter focuses on improving the Product Matching module, optimized for recall in semantic matching and precision in ranking.

Current search frameworks, as shown in Figure 3.1, include two major modules for retrieving the product matches for a given input query [96]; (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 [81, 163] 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 [63, 96] 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 [8] 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 [45].

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., $P(nike\ adidas) = P(nike \cup adidas)P(nike)P(adidas)$.

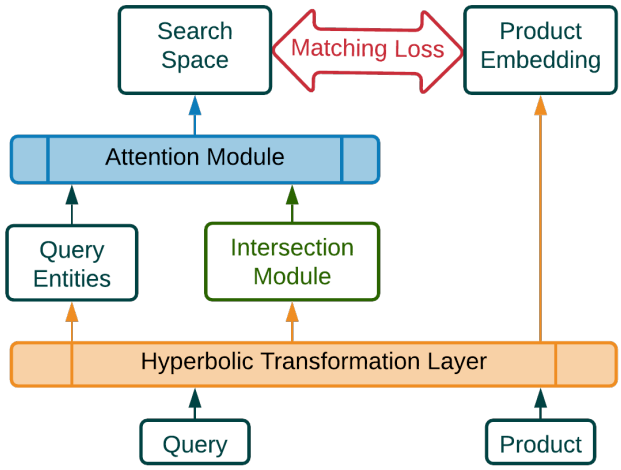


Figure 3.2: Overview of the proposed ANTHEM model. The query entities are encoded in the hyperbolic space as hyperboloids. Attention over the individual entities and their intersections results in the final search space which is matched against the product embeddings.

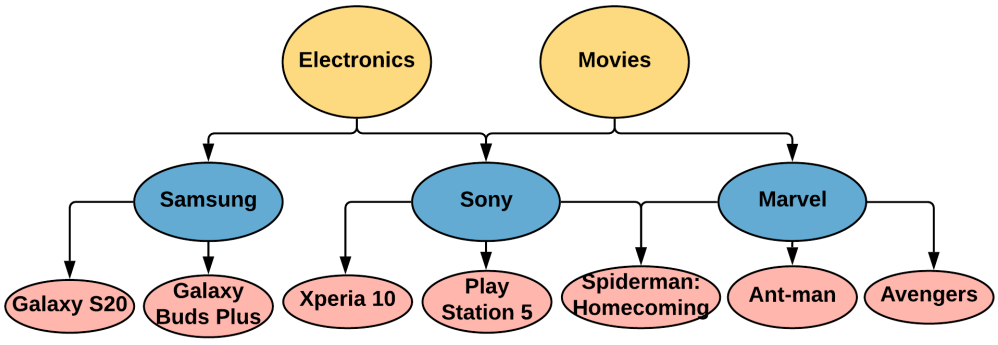


Figure 3.3: Hierarchy of products in the catalogue.

To alleviate the above challenges, we propose AtteNTive Hyperbolic Entity Model (ANTHEM), illustrated in Figure 3.2, 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 3.3) 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 [26, 45, 118]. 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 chapter 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 chapter is organized as follows: Section 3.2 discusses the relevant background. Section 3.3 formulates the product search problem and explains the proposed ANTHEM framework. In Section 3.6, we describe the real-world e-commerce datasets, state-of-the-art baselines and evaluation metrics used to evaluate the proposed model. Section 3.9 concludes the chapter.

3.2 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.

3.2.1 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 [1, 163] 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 [109]. DESM model [89] successfully leverages Word2Vec [82] vectors to rank documents for web search. Furthermore, Diaz et al. [36] 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 [63] 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 [59] and R-DSSM [98] 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 [51] which utilizes a vocabulary interaction matrix in order to capture local semantic information. MatchPyramid [99] extends this approach further by using a convolution operation over the interaction matrix. DUET [84]

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. [96] 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. [53] and Wang et al. [131] use grid-based search and meta-learning to improve the search experience. Nguyen et al. [93] use an adversarial model to learn hard-to-classify query-product pairs and Li et al. [75] improve product search by aggregating information from multiple languages.

3.2.2 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. [45] 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 ($\odot_c$).

$$g_x^{\mathbb{H}} \coloneqq \lambda_x^2 \; g^{\mathbb{E}} \quad \text{where } \lambda_x \coloneqq \frac{2}{1-\|x\|^2} \tag{3.1}$$

$$x \oplus_c y \coloneqq \frac{(1+2c\langle x,y\rangle + c\|y\|^2)\,x + (1-c\|x\|^2)\,y}{1+2c\langle x,y\rangle + c^2\|x\|^2\|y\|^2} \tag{3.2}$$

$$x \ominus_c y \coloneqq x \oplus_c -y \tag{3.3}$$

$$\exp^c(v) \coloneqq \tanh\left(\sqrt{c}\frac{\lambda_0^c\|v\|}{2}\right)\frac{v}{\sqrt{c}\|v\|} \tag{3.4}$$

$$r \odot_c x \coloneqq \exp^c(r log_0^c(x)),\ \forall r \in \mathbb{R}, x \in \mathbb{H}_c^n \tag{3.5}$$

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 [45].

Recently, hyperbolic spaces have been successfully leveraged to learn representations of graph networks [50] and knowledge graphs [18, 133]. Given the hierarchical nature of product catalogue, we extend hyperbolic functions to 2-dimensional geometries in ANTHEM to improve its performance for product search.

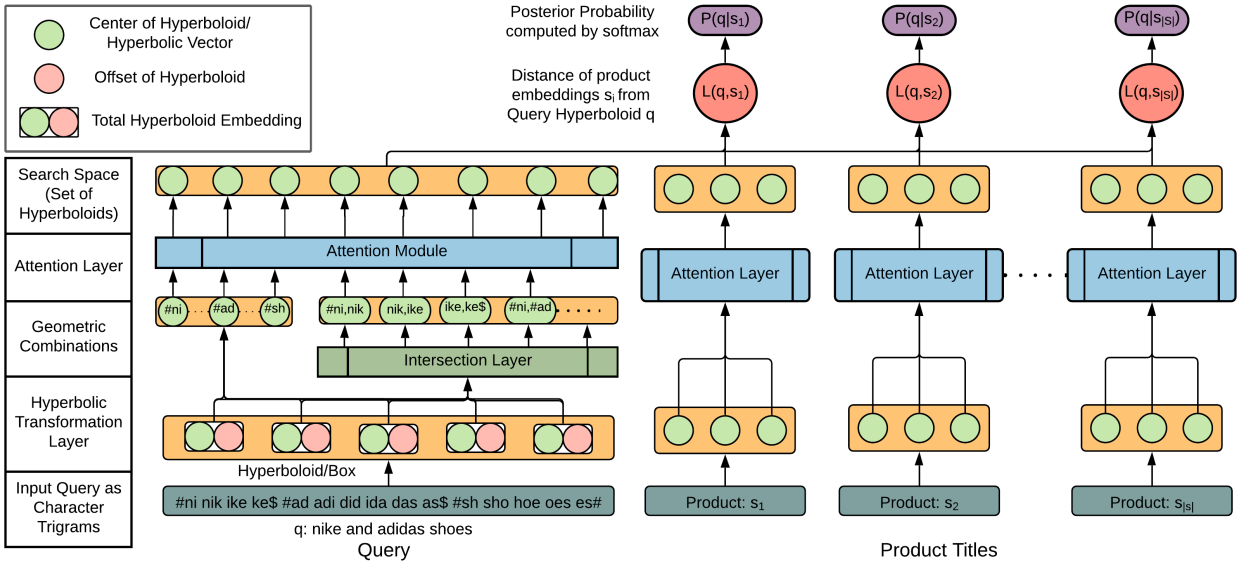


Figure 3.4: The overall architecture of our ANTHEM model. The model encodes the queries' search space as a set of hyperboloids using attention over its entities and inter-entity intersections. The products are encoded as hyperbolic vectors with a self-attention on their char-trigrams. Finally, ANTHEM calculates the distance between product vectors and query hyperboloids and utilizes softmax to output a probability distribution over the products.

3.3 The Proposed ANTHEM Model

In this section, we first discuss the problem setup for product search and query-matching, and then describe the layers and overall architecture of ANTHEM (illustrated in Figure 3.4).

3.3.1 Problem Setup and Notations

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.

Query Matching. Due to the computational intensity of product search, most systems maintain pre-processed results for frequent queries [1]. 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 goal of query-matching is to predict a class y_{ij} for a query-pair (q_i, q_j). To this end, ANTHEM applies the query encoder 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]). The reason for using character trigrams, instead of SentencePiece encoders, is to improve interpretability. The sentence embeddings generated by SentencePiece depend on the language of the input and hence, the interpretability mechanism is not generalizable to different languages, which is necessary for multilingual domains such as e-commerce. 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 $(cen(q_i), lim(q_i)) \in \mathbb{R}^{2d}$:

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

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

3.3.2 Layers of the ANTHEM

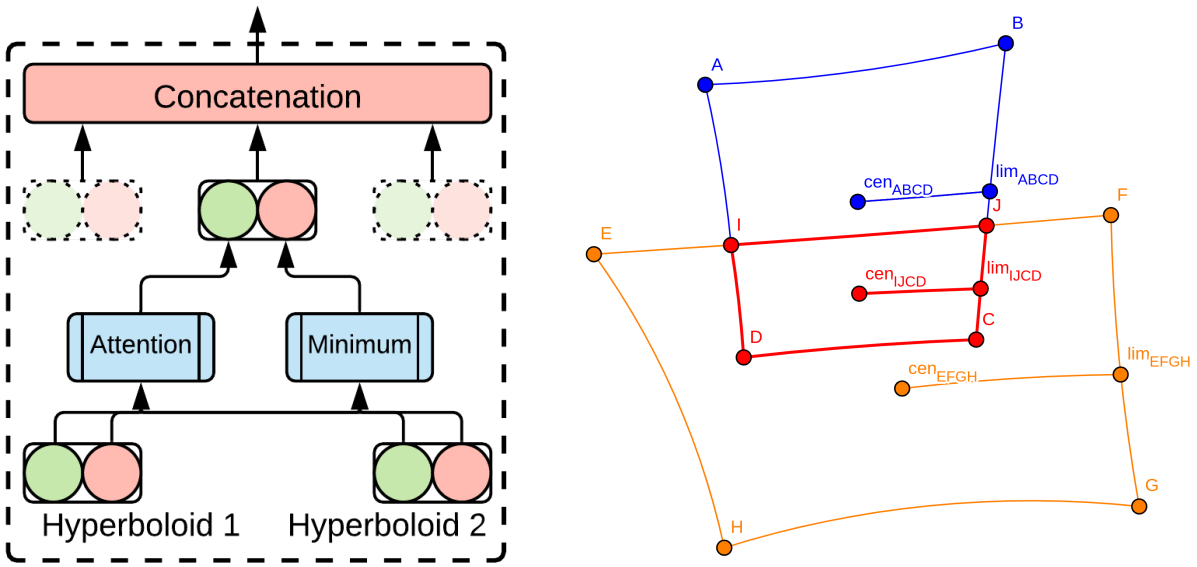
Hyperbolic Transformation Layer

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 $\mathcal{O}(1)$ 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. (3.1). 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.

Intersection Layer



(a) Intersection Layer (b) Intersection of Hyperboloids

Figure 3.5: 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.

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 [104]. We extend the given Euclidean intersection function to the hyperbolic space and define the intersection of queries $q_{ij} = \{q_i, q_j\}$ as $H_{q_{ij}}$:

$$H_{q_{ij}} = \left(cen\left(H_{q_{ij}}\right), lim\left(H_{q_{ij}}\right)\right) \tag{3.6}$$

$$cen\left(H_{q_{ij}}\right) = \sum_{n=i,j} a_n \odot_c cen\left(H_{q_n}\right); \quad a_n = \frac{exp^c(f(H_{q_n}))}{\sum_n exp^c(f(H_{q_n}))} \tag{3.7}$$

$$lim\left(H_{q_{ij}}\right) = Min\left(lim\left(H_{q_i}\right), lim\left(H_{q_j}\right)\right) \tag{3.8}$$

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 3.5) are aggregated by an attention layer [123] 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.

3.3.3 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 [123]. 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}:

$$f_{att}\left(H_i\right) = e_i = \sum_j \frac{exp^c(\alpha_{ij})}{\sum_i exp^c(\alpha_{ij})} H_i; \quad \alpha_{ij} = \frac{H_i^T H_j}{\sqrt{4d}} \tag{3.9}$$

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 [50] (shown in Figure 3.4). For a product s, the product encoder returns a hyperbolic vector $s \in \mathbb{H}^d$. Thus, unlike traditional approaches [63, 96], 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:

$$dist(s, QS(q)) = Min\left(\{d_{hyp}\left(H_i, s\right)\} \forall H_i \in QS(q)\right) \tag{3.10}$$

$$d_{hyp}\left(s, H_i\right) = d_{out}\left(s, H_i\right) \oplus_c \gamma d_{in}\left(s, H_i\right) \tag{3.11}$$

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

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

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

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

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. [26] 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 [47]. 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:

$$\hat{y_i} = P(s_i|q) = \frac{exp^c(dist(s_i, QS))}{\sum_{s_i \in S} exp^c(dist(s_i, QS))} \tag{3.12}$$

$$L(\hat{y}, y) = \left(-\sum_{i=1}^{|S|} y_i \log\left(\hat{y}_i\right) - (1 - y_i) \log\left(1 - \hat{y}_i\right)\right) \tag{3.13}$$

3.3.4 Implementation Details

We implemented ANTHEM using Keras [22] on eight Nvidia V100 GPUs. For gradient descent in hyperbolic space, we adopt Riemannian Adam [4] with an initial learning rate of 0.0001 and standard β values of 0.9 and 0.999 and apply ReLU activation function [88]. The sensitivity of ANTHEM to hyper-parameters is presented in Section 3.4 and its performance with different number of GPUs is provided in Section 3.5. 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 2 provides the pseudo-code for training

Algorithm 2: 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. (3.6)
    $QS = f_{att}(H_q \cup H_{q_\cap})$ using Eq. (3.9)
    # 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. (3.10),(3.12)
    $l = l + L(\hat{y}, y)$; using Eq. (3.13)
    # Update θ with back-propagation
    $\theta \leftarrow \theta - \nabla_\theta l$;
  end
end
return P_θ

ANTHEM[2] on the task of product search. Algorithm 3 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:

$$\hat{y} = \mu\left(\frac{\|QS_{q_i}.QS_{r_j}\|}{\|QS_{q_i}\|\|QS_{r_j}\|}\right) \quad \forall \;\; QS_{q_i} \in QS_q, QS_{r_j} \in QS_r \tag{3.14}$$

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

Algorithm 3: 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. (3.1)
    $H_\cap = \{H_i \cap H_j\} \forall i, j : 1 \rightarrow n; H_i, H_j \in H_q$; via Eq. (3.6)
    $QS_q = f_{att}(H_q \cup H_\cap)$ via Eq. (3.9)
    # 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. (3.13)
    # Update ϕ with back-propagation
    $\phi \leftarrow \phi - \nabla_\phi l$;
  end
end
return X_ϕ

[2]Our code: https://github.com/amazon-research/hyperbolic-embeddings

3.4 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 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 3.7.

In Figure 3.7(a), we observe that ANTHEM is able to converge in under 50 epochs with Riemannian Adam optimizer (details provided in Section 3.3.4). Figure 3.7(b) 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 3.7(c), 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 3.7(d) 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.

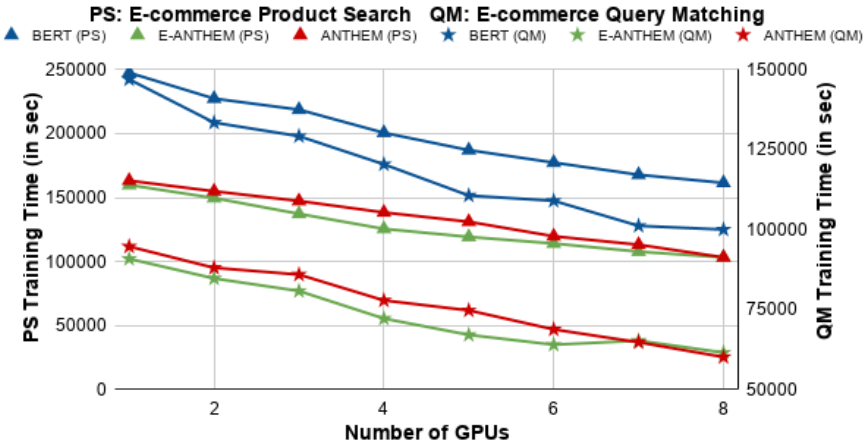
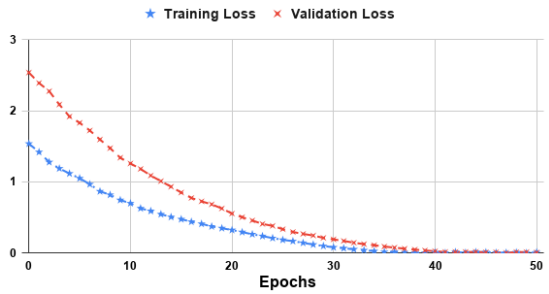


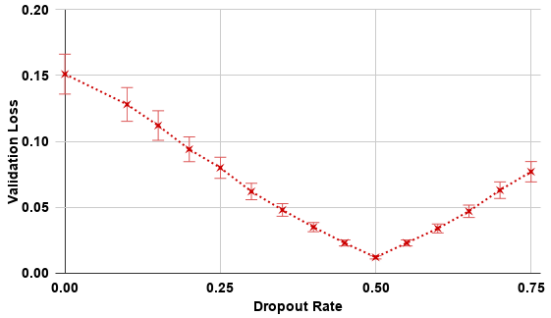
Figure 3.6: Total training time taken by ANTHEM and the best baseline (BERT) model using different number of GPUs (lower is better).

3.5 Number of GPUs

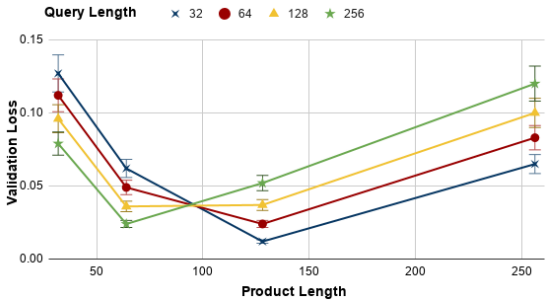
Figure 3.6 shows the dependence of training time on the number of GPUs, which depicts the feasibility of ANTHEM's parallelization. We notice that ANTHEM has a lower training time than the best performing comparison baseline, BERT.



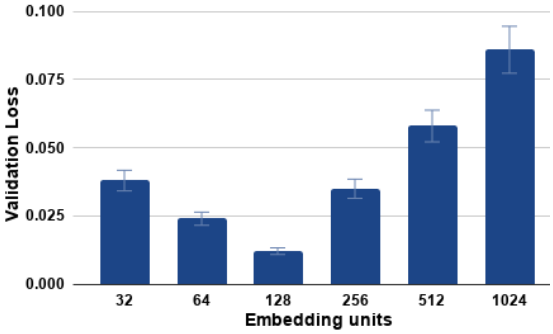
(a) Training and Validation loss across epochs (lower is better).



(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 3.7: Illustration of parameter sensitivity of the proposed ANTHEM model.

3.6 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 chapter.

- **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 3.1: 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

3.6.1 Datasets

We conducted experiments on various product search and query-matching datasets. Table 3.1 presents additional details about the data distribution for these datasets.

- *E-commerce Product Search*[3]: 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[4] or never purchased for a given query. The dataset is completely anonymized, and subsampled to enable efficient model training.
- *Public E-commerce Search Relevance*[5]: 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.

[3]Proprietary dataset
[4]To avoid anomalies, we only consider products purchased more than a predefined number of times.
[5]https://data.world/crowdflower/ecommerce-search-relevance

Table 3.2: 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 $\leq$ **0.05**.

a E-commerce Product Search (in %)

Models	**NDCG@3**	**NDCG@5**	**NDCG@10**	**MAP**	**MRR**
ARC-II	0	0	0	0	0
KNRM	12.5	12.8	15.0	12.8	16.7
DUET	13.1	13.3	15.4	13.4	14.7
DRMM	20.5	22.8	24.4	20.3	21.7
aNMM	21.0	23.9	26.8	20.3	22.9
MatchPyramid	25.6	25.6	26.8	25.0	34.7
C-DSSM	31.3	29.4	44.7	32.6	27.8
MV-LSTM	34.7	33.3	55.7	34.3	37.7
BERT	38.6	37.2	65.9	40.1	51.0
E-ANTHEM	49.4	46.7	66.7	51.2	62.9
ANTHEM	**51.1***	**48.9***	**80.9***	**53.5***	**65.4***

b Public E-commerce Search Relevance (in %)

Models	**NDCG@3**	**NDCG@5**	**NDCG@10**	**MAP**	**MRR**
ARC-II	59.2	58.1	54.4	58.2	48.5
KNRM	66.6	65.5	62.6	65.6	56.6
DUET	66.9	65.8	62.8	66.0	55.6
DRMM	71.3	71.3	67.7	70.0	59.0
aNMM	71.6	72.0	69.0	70.0	59.6
MatchPyramid	74.3	72.9	69.0	72.8	65.3
C-DSSM	77.7	75.2	78.7	77.1	61.9
MV-LSTM	79.7	77.5	84.7	78.2	66.8
BERT	82.1	79.7	90.2	81.5	73.2
E-ANTHEM	88.5	85.2	90.7	88.0	79.0
ANTHEM	**89.5***	**86.5***	**98.4***	**89.3***	**80.2***

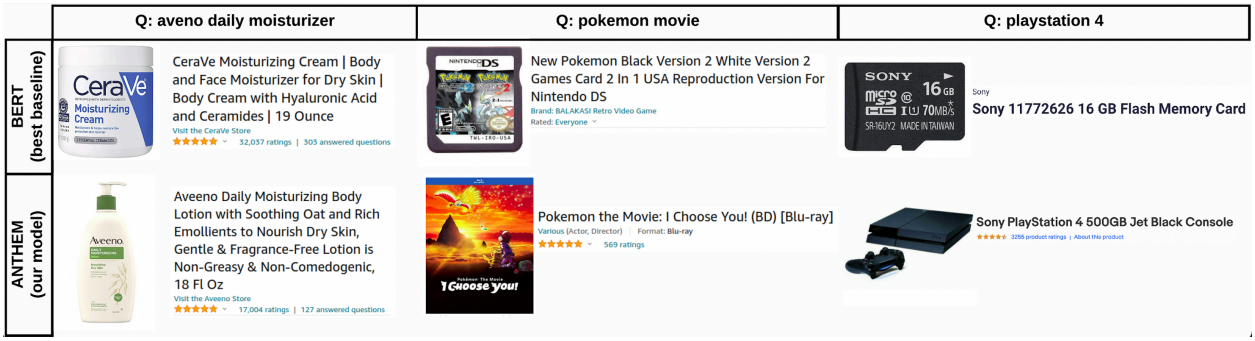


Figure 3.8: Example results for sample queries shown by our model and the best performing baseline. The results given are the third result for the query. The first two results are not shown here because they were equally appropriate for the query and the figure aims to show the differences between BERT and our model.

- *E-commerce Query Matching*[3]: 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*.

3.6.2 Baselines

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

- **ARC-II** [59] utilizes a joint learning Siamese convolutional network to semantically match natural language sentences.
- **KNRM** [139] 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** [51] 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** [143], 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** [99] computes an interaction matrix between queries and document, and then passes it through CNN layers with dynamic pooling to compute sentence similarity.
- **C-DSSM** [109] 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 [96].
- **DUET** [84] combines the semantic and lexical matching strengths of C-DSSM and DRMM, in a deep convolutional architecture, to compute sentence similarity.
- **MV-LSTM** [129] 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** [34] 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 [52] and the hyper-parameters are tuned using grid-search.

3.6.3 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**), and Mean Reciprocal Rank (**MRR**). The datasets are split into training, valid and test sets of ratio 7:1:2, as given in Table 3.1. The results on the test set are presented in Table 3.2.

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 $\approx 9\%$. 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 3.8, 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.

3.6.4 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 3.1. The results are presented in Table 3.3.

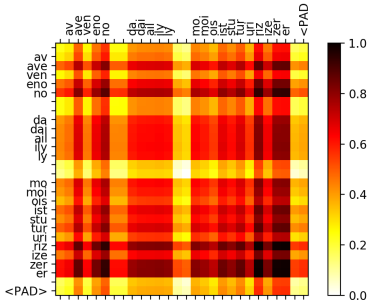
From the results, we observe that ANTHEM is able to outperform the state-of-the-art baselines across datasets by $4\% - 8\%$ in the evaluation metrics. E-ANTHEM, utilizing Euclidean spaces also improves the performance by $3\% - 6\%$. This is empirical evidence that ANTHEM's query encoder is able to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

3.6.5 RQ3: Ablation Study

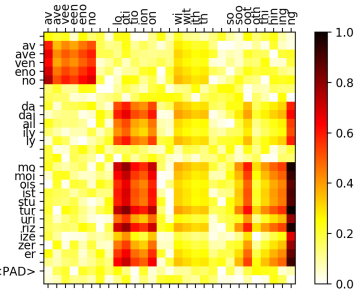
In this experiment, we study the importance of different components that contribute to the overall performance of our proposed model. The components studied in our ablation experiments are: (i) Hyperbolic layer (hierarchical features), (ii) Intersection layer (capturing inter-entity relations) and (iii) Limit parameter (spatial-awareness). The results of our experiments are presented in Table 3.4.

Table 3.3: 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**



(a) Significance of Inter-entity relations analyzed through the attention over Intersection Layer



(b) Significance of query entities to the product entities analyzed through the final attention layer of the product search ANTHEM model.

Figure 3.9: Interpretability Study. We utilize the activations of different attention layers to study the significance of (a)inter-entity relations and (b)the entity itself to the product results.

Table 3.4: 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.

a E-commerce Product Search (in %)

Models	**NDCG@3**	**NDCG@5**	**NDCG@10**	**MAP**	**MRR**
w/o Limit w/o Intersection w/o Hyperbolic	0.0	0.0	0.0	0.0	0.0
w/o Limit w/o Intersection	6.4	6.2	7.5	6.0	7.7
w/o Intersection	23.4	22.9	36.8	24.5	30.2
ANTHEM	41.5	39.6	67.3	43.5	52.6

b Public E-commerce Search Relevance (in %)

Models	**NDCG@3**	**NDCG@5**	**NDCG@10**	**MAP**	**MRR**
w/o Limit w/o Intersection w/o Hyperbolic	41.0	40.9	28.5	39.2	31.2
w/o Limit w/o Intersection	58.0	57.1	47.7	56.3	47.6
w/o Intersection	77.2	75.0	74.2	76.2	67.4
ANTHEM	89.5	86.5	98.4	89.3	80.2

c E-commerce Query Matching (in %)

Models	**Accuracy**	**F-score**	**AUC**
w/o Limit w/o Intersection w/o Hyperbolic	0.0	0.0	0.0
w/o Limit w/o Intersection	4.0	3.7	3.0
w/o Intersection	54.8	54.8	55.2
ANTHEM	104.8	99.3	99.6

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.

3.6.6 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 3.10 which provide a mechanism for researchers to understand the internal functions of our model. ANTHEM is able to match brands to products (Fig. 3.10(a)), translate semantically similar phrases (Fig. 3.10(b)) and leverage hierarchical information for semantic/lexical query-product matching (Fig. 3.10(c)). 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. For a sample query *aveno daily moisturizer*, we observe a significant attention given to brand "aveno", product type "moisturizer" and interaction between entities "aveno" and "moisturizer" which represents an intersection between the two entities (illustrated in Figure 3.9(a)). Additionally, from Figure 3.10, we notice that highest attention is given to maps between "aveno" to "aveeno" and "moisturizer" to "lotion". Hence, from the given activations, we infer that internally ANTHEM is able to establish the translation from "aveno" and "moisturizer" to "aveeno" and "lotion", respectively. Hence, it returns the results from an intersection of brand entity "aveeno" and item entity "moisturizer"; the most relevant being "aveeno lotion with soothing ...".

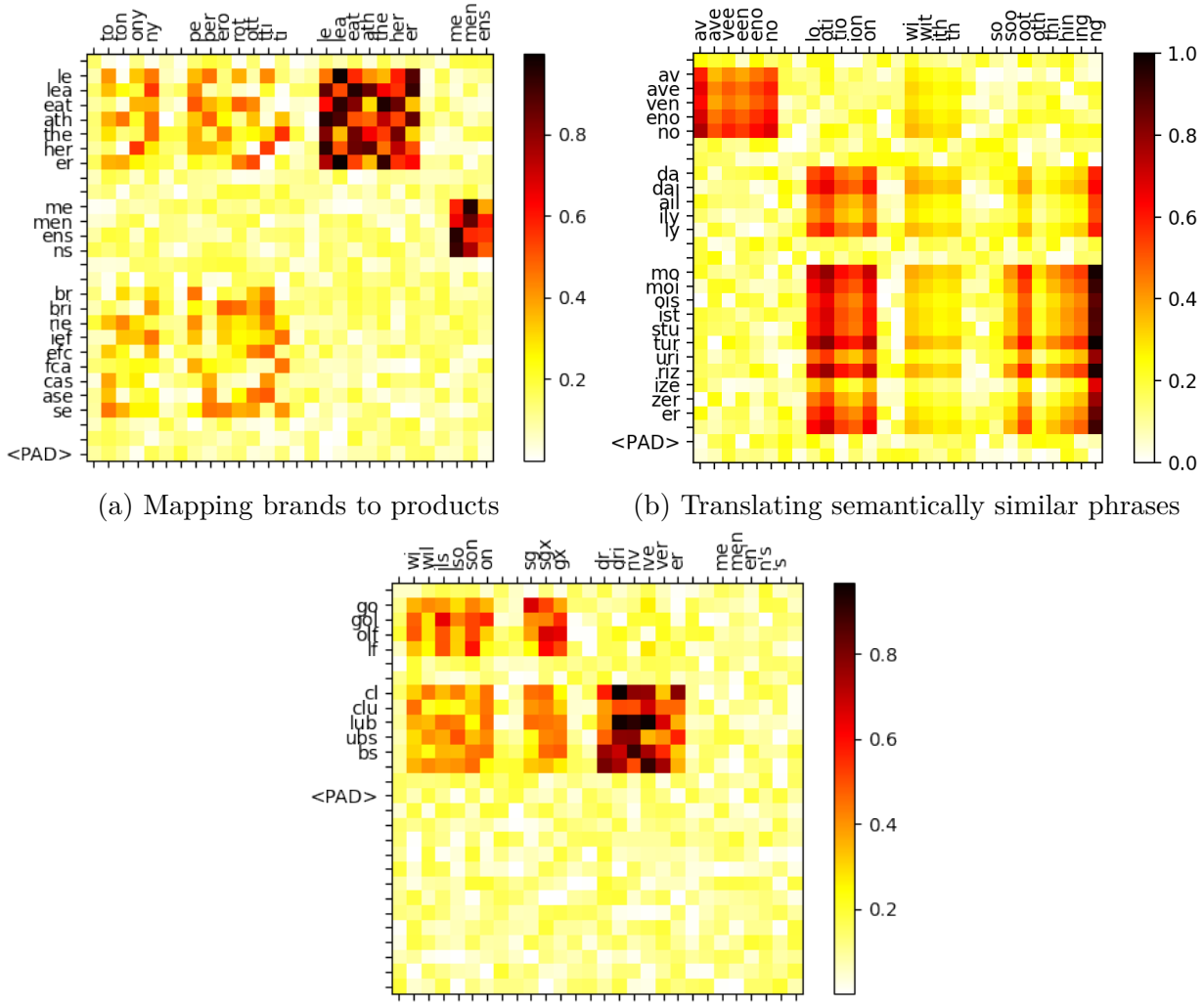
3.7 Deployment

In this section, we discuss the broader impact and deployment strategy to employ ANTHEM in a production environment.

3.7.1 Deployment Strategy

Our experiments suggest that ANTHEM is a state-of-the-art for interpretable product search. Deploying this framework in an online setting will include offline training on large datasets, and the main challenge will be running inference in near real-time.

E-commerce companies have large-scale, deep learning models deployed and running in real-



(a) Mapping brands to products

(b) Translating semantically similar phrases

(c) Semantic/lexical matching

Figure 3.10: 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`.

time [96], and ANTHEM can seamlessly replace these systems. Specifically, the text processing and tokenization components remain the same, and (Euclidean) vector query encoder can be replaced by that of ANTHEM. Learning item representations is an offline step, and these representations can be cached for efficient retrieval. Thus, during runtime, ANTHEM only needs to do a forward pass over the query encoder, and retrieve the cached items by performing the matching in hyperbolic space.

3.7.2 Computational Complexity

To analyze ANTHEM's capability to be used in industry setting, we need to study both its training and inference complexity. The model's parameter study, training time and inference runtime are provided in Table 3.5. 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 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; 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.

3.8 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) 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 customers 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 [68]. In ANTHEM, we aim to provide a possible alternative which considers all possible intents and statistically infers the right intent through the history of product purchases. Additionally, we design our model in a joint learning framework so that it conforms

Table 3.5: 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 3.1. '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
Duet	5,379,580	1,742	300	7.4	1.3
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
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

to existing architectures for easier deployment and is applicable to additional problems such as web search and semantic matching.

3.9 Summary

In this chapter, 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.

Chapter 4

TESH-GCN: Text Enriched Sparse Hyperbolic Graph Convolutional Networks

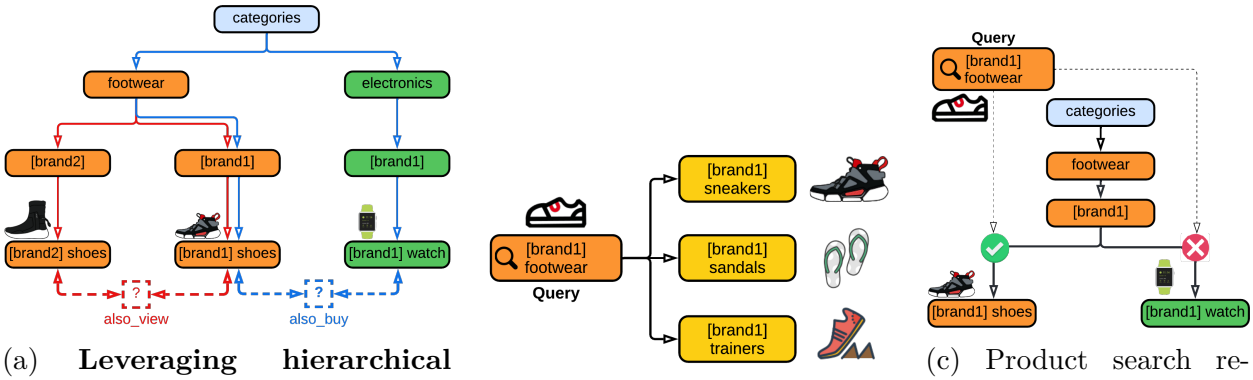
HHeterogeneous networks, which connect informative nodes containing semantic information with different edge types, are routinely used to store and process information in various real-world applications. Graph Neural Networks (GNNs) and their hyperbolic variants provide a promising approach to encode such networks in a low-dimensional latent space through neighborhood aggregation and hierarchical feature extraction, respectively. However, these approaches typically ignore metapath structures and the available semantic information. Furthermore, these approaches are sensitive to the noise present in the training data. To tackle these limitations, in this chapter, we propose Text Enriched Sparse Hyperbolic Graph Convolution Network (TESH-GCN). In TESH-GCN, we use semantic node information to identify relevant nodes and extract their local neighborhood and graph-level metapath features. This is done by applying a reformulated hyperbolic graph convolution layer to the sparse adjacency tensor using the semantic node information as a connection signal. These extracted features in conjunction with semantic features from the language model (for robustness) are used for the final downstream tasks. Experiments on various heterogeneous graph datasets show that our model outperforms the state-of-the-art approaches by a large margin on the task of link prediction. We also report a reduction in both the training time and model parameters compared to the existing hyperbolic approaches through a reformulated hyperbolic graph convolution. Furthermore, we illustrate the robustness of our model by experimenting with different levels of simulated noise in both the graph structure and text, and also, present a mechanism to explain TESH-GCN's prediction by analyzing the extracted metapaths.

4.1 Introduction

Heterogeneous networks, which connect informative nodes containing semantic information with different edge types, are routinely used to store and process information in diverse domains such as e-commerce [28, 29], social networks [24, 73], medicine [25, 31], and citation networks [107]. The importance of these domains and the prevalence of graph datasets link-

ing textual information has resulted in the rise of Graph Neural Networks (GNNs) and their variants. These GNN-based methods aim to learn a node representation as a composition of the representations of nodes in their multi-hop neighborhood, either via random walks [49, 101], neural aggregations [54, 70, 124], or Boolean operations [138]. However, basic GNN models only leverage the structural information from a node's local neighborhood, and thus do not exploit the full extent of the graph structure (i.e., the global context) or the node content. In the context of e-commerce search, based on a consumer's purchase of "[brand1] shoes", it is difficult to identify if they would also purchase "[brand2] shoes" or "[brand1] watch" merely on the basis of the products' nearest graph neighbors, however, global information on purchase behavior could provide additional information in identifying and modeling such purchase patterns. Analysis into such limitations has led to research into several alternatives that capture additional information such as hyperbolic variants [17, 23, 45, 67] to capture the latent hierarchical relations and hybrid models [146, 159] to leverage additional text information from the nodes in the graph. In spite of their preliminary success, these aforementioned techniques fundamentally suffer from several critical limitations such as non-scalability and lack of robustness to noise in real-world graphs when applied in practice. Certain other attempts on aggregating a graph's structural information [148] utilize graph metrics such as centrality encoding and sibling distance to show improved performance over other approaches. However, it is impractical to manually incorporate every one of the exhaustive set of graph metrics available, leaving practitioners to search for a better approach to automatically detect the most relevant graph features that can improve downstream tasks. For instance, metapaths have been identified as heterogeneous paths between different nodes that preserve long-distance relations and have proven to be effective message-passing paths in various graph problems [44]. A metapath is defined as a path in a heterogeneous graph that connects nodes of specific types through intermediate nodes of specific types, where the intermediate nodes act as bridges between the specific types of nodes. For example, consider a heterogeneous graph with nodes representing authors, papers, and venues, where edges between them indicate authorship, paper publication, and paper presentation at a venue. A metapath of length 3 that connects two authors could be $\text{Author} \xrightarrow{\text{Paper}} \xrightarrow{\text{Venue}} \xrightarrow{\text{Paper}} \text{Author}$, where the intermediate nodes are papers and venue. These metapaths can capture long-distance relations between nodes of different types, making them effective for message-passing in graph problems. However, due to computational constraints, metapaths are typically only aggregated locally, meaning that only a local k-hop neighborhood of a node in a heterogeneous graph is considered during the learning process. On the other hand, learning global metapaths could capture long-term relations between the nodes. To learn metapaths, we need to encode the path between two nodes and the semantic information contained in the path. Thus, The adjacency tensor of a heterogeneous graph[1] with a semantic signal can be used to extract both metapath information as well as aggregate local neighborhood features. *Efficiently encoding the entire adjacency tensor in training graph neural models can thus help capture all relevant metapath features.*

[1] for a homogeneous graph, it will be a matrix



(a) **Leveraging hierarchical structures and metapaths** help us distinguish between items that are complementary (also_buy) or alternatives (also_view) of each other.

(b) **Integrating semantic content** with product features allows us to match different products in the catalogue with the query "[brand1] footwear".

(c) Product search requires **robustness to noise** in the hierarchical product graph structure caused by miscategorized items.

Figure 4.1: Challenges of graph representation learning in the E-commerce domain.

In addition to this, the nodes in the graph datasets also contain auxiliary information in different modalities (generally text) such as product descriptions in e-commerce graphs and article titles in citation networks. Such textual content can be encoded using popular transformer models [34], and consequently serve as an additional source of information. Thus, integrating these transformer models in the graph's representation learning process should improve the nodes' feature content during message aggregation and enhance the node representations. Recent hybrid graph-text based techniques [146, 159] also attempt to integrate the node representations with semantic embeddings by initializing the node features with fixed pre-processed semantic embeddings. But, this does not completely leverage the representational power of transformer networks which can learn the task-specific semantic embeddings. Hence, we require a better approach that is able to focus both on the graph and text representation learning towards the downstream task. To summarize, in this chapter, we aim to create a unified graph representation learning methodology that tackles the following challenges (examples from the e-commerce domain given in Figure 4.1):

1. *Leveraging metapath structures:* Existing GNN frameworks aggregate information only from a local neighborhood of the graph and do not possess the ability to aggregate graph-level metapath structures. However, graph-level information can aid in several graph analysis tasks where node's local neighborhood information is insufficient, e.g., in Figure 4.1(a), we note that local node-level information is unable to distinguish between the relations of "also_buy" and "also_view", whereas, graph-level information allows us to do make the differentiation. Indeed, when attempting to combine information from the entire graph, existing methods suffer from over-smoothness [97]. Moreover, the size of modern graph datasets renders aggregating information from the full graph infeasible.

2. *Incorporating hierarchical structures:* Most of the real-world graphs have inherent hierarchies, which are best represented in a hyperbolic space (rather than the traditional Euclidean space), e.g., the product hierarchy shown in Figure 4.1(a). However, existing hyperbolic GNNs [17, 45] do not leverage the full graph when aggregating information due to both mathematical and computational challenges.

3. *Integrating textual (semantic) content:* Previous methods for integrating semantic information of the nodes are relatively ad-hoc in nature. For example, they initialize their node representations with text embeddings for message aggregation in the GNNs [159]. Such methods fix the semantic features and do not allow the framework to learn task-specific embeddings directly from the nodes' original content, e.g., in Figure 4.1(b), the product tokens "sneakers" and "sandals" are closer to the query token "footwear" in the e-commerce domain which is not the case in a broader semantic context.

4. *Robustness to noise:* Real-world graphs are susceptible to noise and hence require robust graph representation learning mechanisms, especially in the presence of multiple forms of data (i.e., graph structure and textual content), e.g., in Figure 4.1(c), we observe that the task of product search is susceptible to noise in the product catalogue due to miscategorized items. Previous approaches do not leverage the complementary nature of graphs and text to improve robustness to noise in both of these modalities.

To tackle the above challenges, we introduce *Text Enriched Sparse Hyperbolic Graph Convolution Network* (TESH-GCN), a novel architecture towards learning graph representations (illustrated in Figure 4.2) for the task of link prediction. In the case of heterogeneous graphs, the node adjacency information can be modeled as a tensor and can be used to both aggregate local neighborhood as well as extract graph-level metapath structures [44]. However, real-world adjacency tensors are extremely sparse ($\approx 99.9\%$ entries are zero)[2]. *TESH-GCN leverages the sparsity to efficiently encode the entire adjacency tensor and automatically captures all the relevant metapath structures.* We also utilize dense semantic signals from the input nodes which improve the model's robustness by making the representations conditional on both the graph and text information. To capture the semantic information of the nodes, we leverage the recent advances in language models [34, 42] and jointly integrate the essential components with the above mentioned graph learning schemes. This allows nodes' feature content to be passed through the message aggregation and enhance performance on downstream tasks. In addition to this, our model's attention flow enables the extraction and comprehension of weighted inter-node metapaths that result in the final prediction. Summarizing, following are the major contributions of this chapter:

1. We introduce Text Enriched Sparse Hyperbolic Graph Convolution Network (TESH-GCN), which utilizes semantic signals from input nodes to extract the local neighborhood and metapath structures from the adjacency tensor of the entire graph to aid the prediction task.

[2]Sparsity ratios of our datasets are given in Table 4.2.

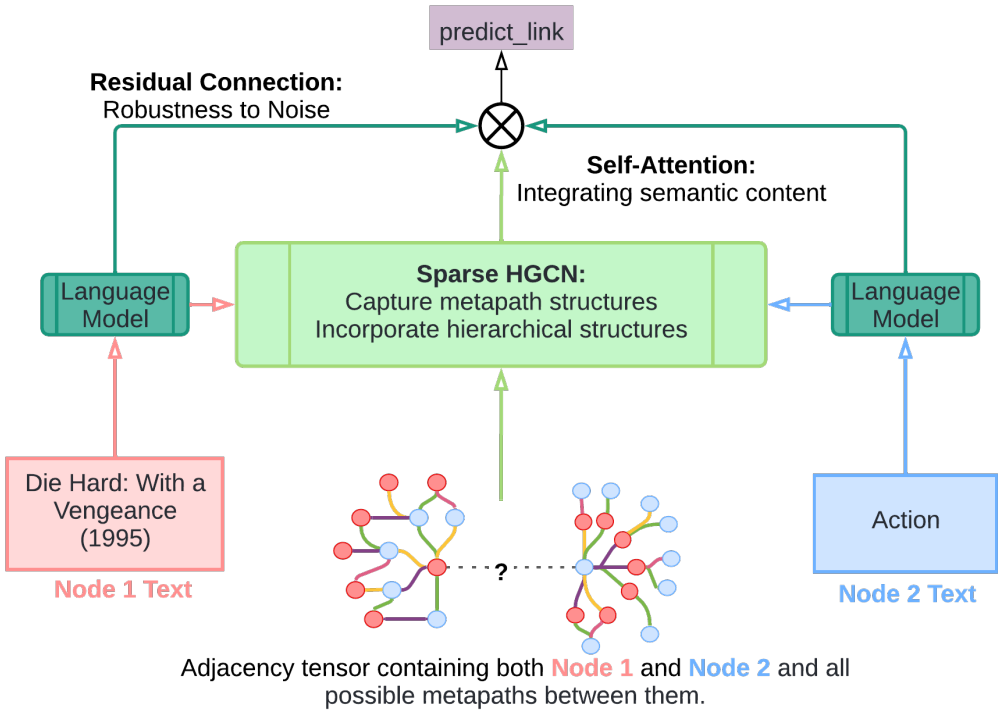


Figure 4.2: An overview of the proposed TESH-GCN model. The semantic signals are efficiently integrated with the nodes' local neighborhood and metapath structures extracted from the adjacency tensor.

2. To enable the coordination between semantic signals and sparse adjacency tensor, we reformulate the hyperbolic graph convolution to a linear operation that is able to leverage the sparsity of adjacency tensors to reduce the number of model parameters, training and inference times (in practice, for a graph with 10^5 nodes and 10^{-4} sparsity this reduces the memory consumption from 80GB to 1MB). To the best of our knowledge, no other method has utilized the nodes' semantic signals to extract both local neighborhood and metapath features.

3. Our unique integration mechanism, not only captures both graph and text information in TESH-GCN, but also, provides robustness against noise in the individual modalities.

4. We conduct extensive experiments on a diverse set of graphs to compare the performance of our model against the state-of-the-art approaches on link prediction and also provide an explainability method to better understand the internal workings of our model using the aggregations in the sequential hyperbolic graph convolution layers.

The rest of this chapter is organized as follows: Section 4.2 discusses the related work in the areas of link prediction and hyperbolic networks. Section 4.3 describes the problem statement and the proposed TESH-GCN model. In Section 4.4, we describe the experimental setup, including the datasets used for evaluation, baseline methods, and the performance metrics used to validate our model. Finally, Section 4.5 summarizes the chapter.

4.2 Related Work

In this section, we describe earlier works related to our proposed model, primarily in the context of graph representation learning and hyperbolic networks.

4.2.1 Graph Representation Learning

Early research on graph representations relied on learning effective node representations, primarily, through two broad methods, namely, matrix factorization and random walks. In matrix factorization based approaches [15], the sparse graph adjacency matrix A is factorized into low-dimensional dense matrix L such that the information loss $\|L^T L - A\|$ is minimized. In the random walk based approaches [49, 91, 101], a node's neighborhood is collected with random walks through its edges, and the neighborhood is used to predict the node's representation in a dense network framework. Earlier methods such as LINE [119] and SDNE [130] use first-order (nodes connected by an edge) and second-order (nodes with similar neighborhood) proximity to learn the node representations. These methods form a vector space model for graphs and have shown some preliminary success. However, they are node-specific and do not consider the neighborhood information of a node or the overall graph structure.

In more recent works, aggregating information from a nodes' neighborhood is explored using the neural network models. Graph neural networks (GNN) [106], typically applied to node classification, aggregate information from a nodes' neighborhood to predict the label for the root node. Several approaches based on different neural network architectures for neighborhood aggregation have been developed in recent years and some of the popular ones include GraphSage [54] (LSTM), Graph Convolution Networks (GCN) [70], and Graph Attention Networks (GAT) [124]. Another line of work specifically tailored for heterogeneous graphs [44, 61, 135, 144, 149], utilizes the rich relational information through metapath aggregation. These approaches, while efficient at aggregating neighborhood information, *do not consider the node's semantic attributes or the global graph structure.* In addition, there have been recent efforts to develop application-specific approaches such as OntoProtein [155] for protein ontologies and QA-GNN [147] for question-answering knowledge graphs. These approaches use semantic node embeddings to enhance KG reasoning algorithms. However, they are limited to considering only the local node neighborhoods due to their reasoning objective and are not suitable for capturing global structural information. In the proposed TESH-GCN model, we aim to utilize the node's semantic signal, in congruence with global adjacency tensor, to capture both the node's semantic attributes and its position in the overall graph structure.

4.2.2 Hyperbolic Networks

In recent research [45], graph datasets have been shown to possess an inherent hierarchy between nodes thus demonstrating a non-Euclidean geometry. In [45], the authors provide the gyrovector space model including the hyperbolic variants of the algebraic operations required to design neural networks. The algebraic operations for the Poincaré ball of curvature c are the following: Möbius addition ($\oplus_c$), exponential map ($\exp_x^c$), logarithmic map ($\log_x^c$), Möbius scalar multiplication ($\otimes_c$), and hyperbolic activation (σ_c).

$$
\begin{aligned}
x \oplus_c y &= \frac{(1 + 2c\langle x, y\rangle + c\|y\|^2)\, x + (1 - c\|x\|^2)\, y}{1 + 2c\langle x, y\rangle + c^2\|x\|^2\|y\|^2} \\
\exp_x^c(v) &= x \oplus_c \left(\tanh\left(\sqrt{c}\frac{\lambda_x^c \|v\|}{2} \right) \frac{v}{\sqrt{c}\|v\|} \right) \\
\log_x^c(y) &= \frac{2}{\sqrt{c}\lambda_x^c} \tanh^{-1}\left(\sqrt{c}\| -x \oplus_c y\| \right) \frac{-x \oplus_c y}{\| -x \oplus_c y\|}
\end{aligned}
$$

$$
\begin{aligned}
r \otimes_c x &= \exp_0^c(r log_0^c(x)),\ \forall r \in \mathbb{R}, x \in \mathbb{H}_c^n \\
\sigma_c(x) &= \exp_0^c(\sigma(\log_0^c(x)))
\end{aligned} \tag{4.1}
$$

where $\lambda_x^c = \frac{2}{(1-c\|x\|^2)}$ is the metric conformal factor. Based on these approaches, hyperbolic networks such as HGNN [45], HGCN [17], HAN [50], and HypE [26] have shown to outperform their Euclidean counterparts on graph datasets. However, these approaches still

focus on the nodes' local neighborhood and not the overall graph structure. Furthermore, hyperbolic transformations are performed on entire vectors and are thus inefficient on sparse tensors. In our model, we utilize the $\beta-$split and $\beta-$concatenation operations [111] to optimize the hyperbolic graph convolution for sparse adjacency tensors.

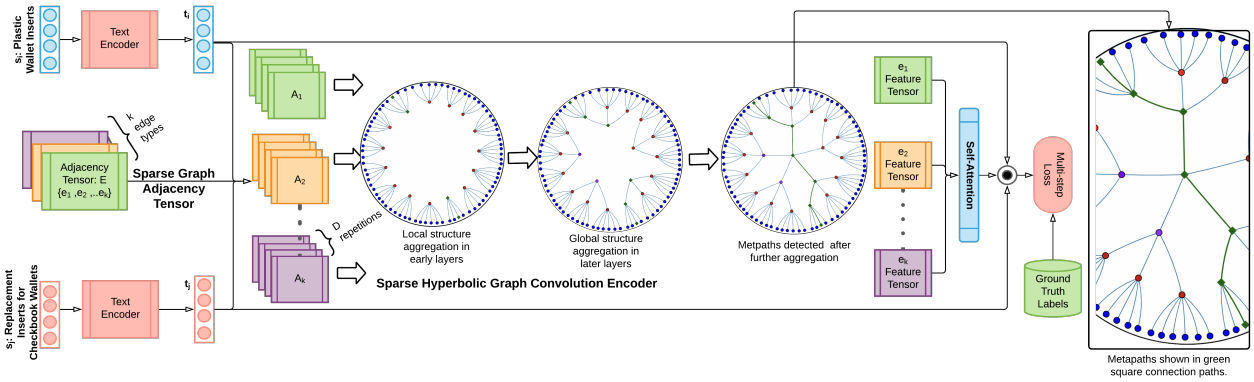


Figure 4.3: Architecture of our proposed model. The Hyperbolic Graph Convolution Encoder aggregates local features in the early layers and global features in the later layers. The encoder also handles sparsity to reduce both time and space complexity.

4.3 The Proposed model

In this section, we first describe the problem setup for link prediction on sparse heterogeneous graphs.[3] We then provide a detailed explanation of the different components of the proposed model and their functionality in the context of link prediction. The overall architecture is depicted in Figure 4.3. The notations used in this chapter are provided in Table 4.1.

4.3.1 Problem Setup

Let us consider a heterogeneous graph $\mathcal{G} = (V, E)$ with K edge types, where $v \in V$ is the set of its nodes and $e_k(v_i, v_j) \in E \in \mathbb{B}^{K\times|V|\times|V|}$ is a sparse Boolean adjacency tensor (which indicates if edge type e_k exists between nodes v_i and v_j or not). Each node v_i also contains a corresponding text sequence s_i. The sparsity of the adjacency tensor and hierarchy of the graph $\mathcal{G}$ is quantified by the sparsity ratio (R, Definition 4.1) and hyperbolicity (δ, Definition 4.3), respectively. Higher sparsity ratio implies that E is sparser, whereas lower hyperbolicity implies $\mathcal{G}$ has more hierarchical relations.

[3]Note that we use link prediction as a running example in this chapter. Other tasks (node/graph classification) can be easily performed by changing the loss function.

Table 4.1: Notations used in the chapter.

Notation	Description
$\mathcal{G}$	the heterogeneous graph
V	set of nodes in graph $\mathcal{G}$
K	number of edge types in the graph $\mathcal{G}$
E	$K \times \|V\| \times \|V\|$-sized boolean adjacency tensor
e_k	$\|V\| \times \|V\|$-sized adjacency matrix edge of type k in E
$e_k(v_i, v_j)$	boolean indicator of edge type k between nodes v_i and v_j
R	sparsity ratio
$\delta(\mathcal{G})$	hyperbolicity of graph $\mathcal{G}$
P_θ	model with parameters θ
y_k	probability that input sample belongs to class k
s_i	textual tokens of node v_i
$LM(x)$	D-sized vector from language model LM of textual tokens x
t_i	D-sized encoded text vector of tokens s_i
A_k	$D \times \|V\| \times \|V\|$-sized stack of adjacency matrix e_k
$W_{f,l}$	filter weights for feature transformation in l^{th} layer
$o_{p,l}$	output of feature transformation in l^{th} layer
α_p	attention weights for feature aggregation in the l^{th} layer
$a_{p,l}$	output scaled by α_p in the l^{th} layer
$h_{p,l}$	final output of the l^{th} convolution layer
α_k	attention weight of the encoding k^{th} adjacency matrix
$h_{k,L}$	attention scaled encoding of the k^{th} adjacency matrix
h_L	output of the sparse hyperbolic convolution layers
$out(A)$	final output of TESH-GCN for input adjacency tensor A
$\hat{y_k}$	ground truth labels of edge type k
$L(y_k, \hat{y_k})$	cross-entropy loss over $\hat{y_k}$ and y_k

Definition 4.1. Sparsity ratio (R) is defined as the ratio of the number of zero elements to the total number of elements in the adjacency tensor;

$$R = \frac{|e_k(v_i, v_j) = 0|}{|V|^2} \tag{4.2}$$

Definition 4.2. Homogenized Graph: Let $\mathcal{G} = (V, E)$ be a heterogeneous graph with a set of nodes V and a set of edges E with K edge types. The homogenized version of $\mathcal{G}$, denoted as $\mathcal{G}'$, is obtained by replacing each edge type $e_k \in E$ to common edge type e using the mapping function $f(e_k) = e \; \forall e_k \in E$.

The edge distance between nodes a and b in the homogenized graph $\mathcal{G}'$, denoted by $dist(a, b)$, is defined as the shortest path length between a and b in the transformed graph. Here, the length of a path is defined as the number of edges along the path $a \to b$.

Definition 4.3. For a graph $\mathcal{G}$, the hyperbolicity (δ) is calculated as described in [48]. Let us say $(a, b, c, d) \in \mathcal{G}$ is a set of vertices, and $dist(a, b)$ indicates the edge distance between vertices a and b in a homogenized version of graph $\mathcal{G}$ (defined in Def. 4.2). Let us define S_1, S_2 and S_3 as:

$$\begin{aligned} S_1 &= dist(a, b) + dist(d, c) \\ S_2 &= dist(a, c) + dist(b, d) \\ S_3 &= dist(a, d) + dist(b, c) \end{aligned}$$

Let M_1 and M_2 be the two largest values in (S_1, S_2, S_3), then $H(a, b, c, d) = M_1 - M_2$ and $\delta(\mathcal{G})$ is given by:

$$\delta(\mathcal{G}) = \frac{1}{2} \max_{(a,b,c,d) \in \mathcal{G}} H(a, b, c, d)$$

For the task of link prediction, given input nodes v_i and v_j with corresponding text sequence s_i and s_j, respectively and an incomplete training adjacency tensor E, our goal is to train TESH-GCN to optimize a predictor P_θ parameterized by θ such that;

$$\begin{aligned} y_k &= P_\theta(z = 1|I) P_\theta(y = k|I), \text{ where } I = \{v_i, v_j, s_i, s_j, E\}, \\ \theta &= \arg\min_\theta \left(- \sum_{k=1}^{K} \hat{y_k} \log (y_k) \right) \end{aligned}$$

where z is a Boolean indicator that indicates if an edge between the two nodes exists ($z = 1$) or not ($z = 0$) and y is a class predictor for each $k \in K$ edge types. $\hat{y_k}$ is the probability of each class $k \in K$ predicted by TESH-GCN and y_k is the ground truth class label.

4.3.2 Text Enriched Sparse Hyperbolic GCN

In this section, we describe the message aggregation framework of TESH-GCN, which allows us to aggregate the node's text-enriched local neighborhood and long metapath features (through semantic signals and reformulated hyperbolic graph convolution) from sparse adjacency tensors in the hyperbolic space. In this section, we detail the (i) methodology of integrating semantic features with graph tensors, (ii) sparse HGCN layer to encode hierarchical and graph structure information efficiently, and (iii) aggregation through self-attention to improve model robustness.

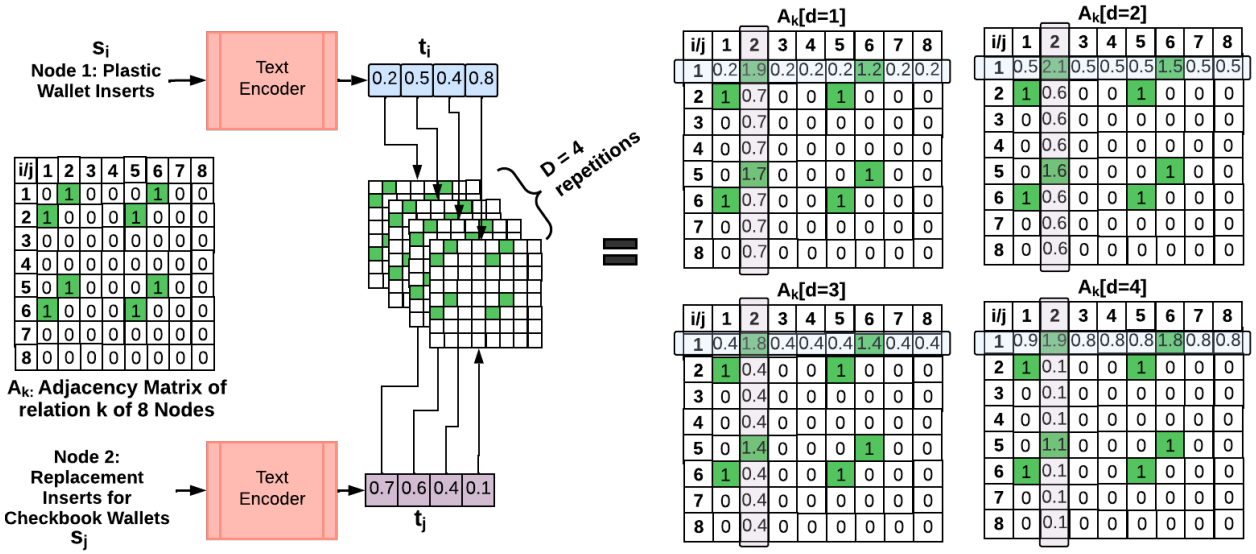


Figure 4.4: Adding semantic signals $t_i, t_j \in \mathbb{R}^{D=4}$ of nodes i and j to the sparse adjacency matrix $A_k \in \mathbb{R}^{64}$ of a graph with $|V| = 8$ nodes and $|E| = 8$ edges. The nodes' independent semantic dimensions are added to their corresponding position in the independent adjacency matrix copies. This addition focuses the subsequent convolution operation on the highlighted areas (due to the presence of non-zeros) to initiate the extraction of graph features at the location of the input nodes.

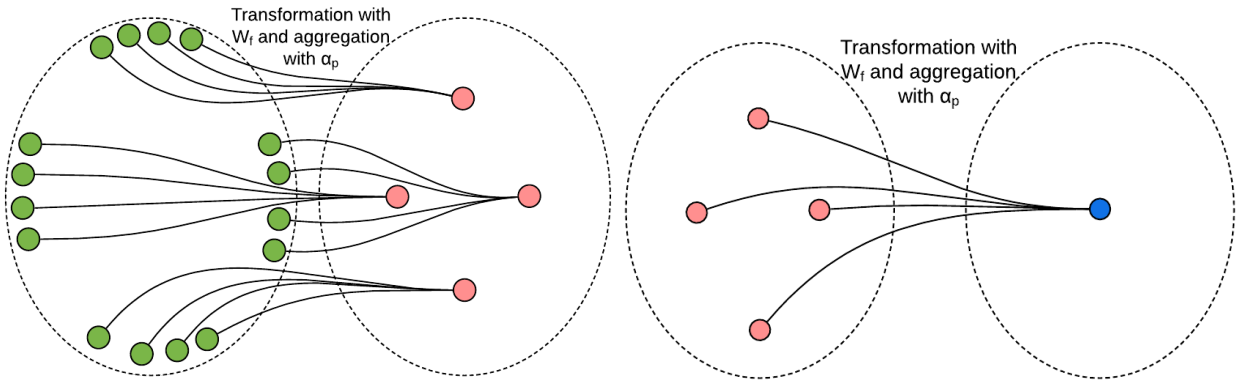
Incorporating Semantics into Adjacency Tensor: To integrate the nodes' textual information with the graph structure, we enhance the adjacency tensor of the heterogeneous graph with semantic features of nodes. We obtain semantic signals using a pre-trained language model (LM) developed by Song et al. (2020) to encode each node's textual data into a vector $t \in \mathbb{R}^D$. It should be noted that the dimensions of a semantic vector are linearly independent and hence, each dimension corresponds to a unique independent semantic feature. Due to this, we cannot simply add the semantic vectors to the adjacency matrix to incorporate them. To overcome this issue, we propose a novel solution wherein we stack D-repetitions of the adjacency matrix e_k to form tensor A_k and add each independent semantic dimension $t[d] \in t$ to a corresponding adjacency matrix $A_k[d] \in A_k$. Moreover, we add the

semantic dimension in the node's position within the adjacency matrix to maintain positional consistency. This ensures that the adjacency tensor A_k captures the nodes' semantic signals in their appropriate locations within the graph structure. Please refer to Figure 4.4 for a clear illustration of the entire process. An important consideration in this operation is that it does increase the density of the adjacency matrix by $\frac{2}{|V|}$. However, we observe that this increase has negligible impact on the sparsity of real-world datasets (statistics provided in Table 4.2).

$$t_i = LM(s_i), \quad t_j = LM(s_j) \tag{4.3}$$

$$A_k[d, i, :] = A_k[d, i, :] + t_i[d], \quad A_k[d, :, j] = A_k[d, :, j] + t_j[d] \quad \forall d = 1 \rightarrow D \tag{4.4}$$

where $A_k[d, i, :]$ represents the i^{th} row in the d^{th} matrix of A_k and $A_k[d, :, j]$ represents the j^{th} column in the d^{th} matrix of A_k. $t_i[d]$ and $t_j[d]$ are the d^{th} dimension of their respective semantic signals. The update operations given above ensure that the adjacency tensor A_k contains information on the semantic signals at the appropriate position in the graph structure. Thus, an efficient encoding of A_k allows us to capture the underlying nodes' structural information and semantic content. We achieve this through the sparse HGCN layer.



(a) Early neighbor aggregation (b) Later metapath aggregation

Figure 4.5: Interpretation of the hyperbolic graph convolution. The first few layers aggregate neighborhood information and the later layers aggregate graph-level metapath information. Darker cells indicate higher weight values.

Sparse Hyperbolic Graph Convolution: To encode the graph structure and latent hierarchy, we need to leverage the adjacency tensor's sparsity in the hyperbolic space for computational efficiency. To achieve this, we reformulate the hyperbolic graph convolution in the following manner. The graph convolution layer has two operations, namely, feature transformation and aggregation, which are achieved through convolution with a filter map of trainable curvature and pooling, respectively. For a matrix of size $m_r \times m_c$ and filter map $f \times f$, graph convolution requires $\approx (m_r - f) \times (m_c - f)$ operations. However, given the high sparsity of adjacency matrices, operations on zero-valued cells will return zero gradients and,

thus not contribute to the learning process. Hence, we only apply the filter transformation to adjacency tensor cells with nonzero values and ignore the zero-valued cells. For the d^{th} input adjacency matrix with elements $x \in A_k[d]$,

$$o_{p,l} = W_{f,l} \otimes_{c_l} x_{p,l-1} \oplus_{c_l} b_l \quad \forall x_{p,l-1} \neq 0 \tag{4.5}$$

$$a_{p,l} = exp^{c_l}_{x_{p,l-1}} \left(\frac{\alpha_p \log^{c_l}_{x_{p,l-1}}(o_{p,l})}{\sum_p \alpha_p \log^{c_l}_{x_{p,l-1}}(o_{p,l})} \right) \tag{4.6}$$

$$h_{p,l} = \sigma_{c_l}(a_{p,l}) \tag{4.7}$$

where $o_{p,l}$ represents the output of feature transformation at the layer l for non-zero input elements $x_{p,l-1}$ of previous layer's $l-1$ adjacency tensor with learnable feature map $W_{f,l}$. c_l and b_l represent the Poincaré ball's curvature and bias at layer l, respectively. $\otimes_{c_l}$ and $\oplus_{c_l}$ are the Möbius operations of addition and scalar multiplication, respectively. $a_{p,l}$ is the output of the scalar-attention [123] over the outputs with attention weights α_p and $h_{p,l}$ is the layer's output after non-linear hyperbolic activation. The initial layers aggregate the sparse neighborhoods into denser cells. As the adjacency tensors progress through the layers, the features are always of a lower resolution than the previous layer (aggregation over aggregation), and thus aggregation in the later layers results in graph-level metapath features, as depicted in Figure 4.5. Note that the computational complexity of calculating $o_{p,l}$ in sparse graph convolutions is $\mathcal{O}(V^2(1-R))$ when compared to $\mathcal{O}(V^2)$ of dense graph convolutions[4]. This indicates a reduction in the total number of computations by a factor of $(1-R) \approx 10^{-4}$. Prior hyperbolic approaches could not utilize sparse convolutions because the hyperbolic operation could not be performed on splits of the adjacency tensor but we enable this optimization in TESH-GCN through the operations of β-split and β-concatenation [111], formulated in Definition 4.4 and 4.5.

Let us say, the d-dimensional hyperbolic vector in Poincaré ball of curvature c is $x \in \mathbb{H}^d_c$ and $\beta_d = B\left(\frac{d}{2}, \frac{1}{2}\right)$ is a scalar beta coefficient, where B is the beta function. Then, the *β-split* and *β-concatenation* are defined as follows.

Definition 4.4. *β-split*: The hyperbolic vector is split in the tangent space with integer length $d_i : \Sigma_{i=1}^{D} d_i = d$ as $x \mapsto v = log^c_0(x) = (v_1 \in \mathbb{R}^{d_1}, ..., v_D \in \mathbb{R}^{d_D})$. Post-operation, the vectors are transformed back to the hyperbolic space as $v \mapsto y_i = exp^c(\beta_{d_i}\beta_d^{-1}v_i)$.

Definition 4.5. *β-concatenation*: The hyperbolic vectors to be concatenated are transformed to the tangent space, concatenated and scaled back using the beta coefficients as; $x_i \mapsto v_i = log^c_0(x_i), v := (\beta_d\beta_{d_1}^{-1}v_1, ..., \beta_d\beta_{d_D}^{-1}v_D) \mapsto y = exp^c(v)$.

[4]Practically, for a graph with 10^5 nodes and a sparsity of 10^{-4}, dense graph convolution requires 80GB of memory (assuming double precision) for one layer, whereas, sparse graph convolution only requires 1MB of memory for the same. This allows us to utilize the entire adjacency tensor, while previous approaches can only rely on the local neighborhood.

The final encoding of an adjacency tensor A_k is, thus, the output features of the last convolution layer transformed to the tangent space with the logarithmic map $h_{k,L} = log_0^{c_L}(h_{k,L})$[5].

Aggregation through Self-Attention: Given the encoding of adjacency tensor of all edge types $A_k \in A$, we aggregate the adjacency tensors such that we capture their inter-edge type relations and also condition our prediction on both the graph and text for robustness. To achieve this, we pass the adjacency tensor encodings $A_k \in A$ through a layer of self-attention [123] to capture the inter- edge type relations through attention weights. The final encoder output $out(A)$ concatenates the features of adjacency tensor with the semantic embeddings to add conditionality on both graph and text information.

$$h_{k,L} = \frac{\alpha_k h_{k,L}}{\sum_k \alpha_k h_{k,L}} \tag{4.8}$$

$$h_L = h_{1,L} \odot h_{2,L} \odot \cdots \odot \ h_{k,L} \tag{4.9}$$

$$out(A) = h_L \odot t_i \odot t_j \tag{4.10}$$

where α_k are the attention weights of edge types and h_L are the adjacency tensors' features. The semantic residual network connection sends node signals to the adjacency tensor and also passes information to the multi-step loss function. The balance between semantic residual network and hyperbolic graph convolution leads to robustness against noisy text or graphs (evaluated empirically in Section 4.4.6).

4.3.3 Multi-step Loss

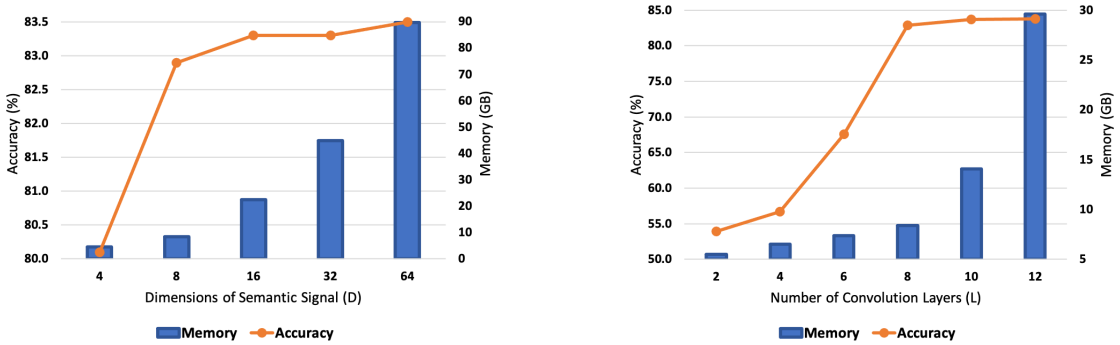
In this work, we consider a generalized link prediction problem in heterogeneous networks where there are two sub-tasks. (i) To predict if a link exists between two nodes and (ii) To predict the class/type of link (if one exists). One method to achieve this goal is to add the non-existence of link as another class. Let us assume we add a class z which indicates the existence of the link ($z = 1$) and $z = 0$ when the link is absent. Then, for the task of link prediction, we need to support the independence assumption, i.e., $z \perp\!\!\!\perp e_k, \quad \forall e_k \in E$, which is not true. Prediction of an edge type e_k is conditional on $z = 1$. Hence, we setup a multi-step loss that first predicts the existence of a link and then classifies it into an edge type.

$$y_k = P_\theta(e_k|x) = P_\theta(z = 1|x)P_\theta(y = e_k|x) \tag{4.11}$$

$$L(y_k, \hat{y_k}) = -\sum_{k=1}^{K} \hat{y_k} \log(y_k) \tag{4.12}$$

[5]The transformation from hyperbolic space to tangent space with logarithmic map is required for attention-based aggregation as such formulation is not well-defined for the hyperbolic space.

where x and θ are the input and model parameters, respectively. L is the cross entropy loss that needs to be minimized. Although we use this generalized link prediction as the task of interest in this chapter, TESH-GCN can be applied to any task such as node/graph classification by replacing the loss with the appropriate loss.



(a) Dimension of semantic signal (D) vs Memory and Accuracy.

(b) No. of graph convolution layers (L) vs Memory and Accuracy.

Figure 4.6: Effect of L and D parameters on memory required and accuracy performance of TESH-GCN on Amazon dataset. Note that we use 16GB of Nvidia V100 GPU for our experiments. For higher than 16GB of memory we place different components on different GPU and moving the tensors among different GPUs adds an insignificant overhead.

4.3.4 Implementation Details

We implemented TESH-GCN using Pytorch [100] on eight NVIDIA V100 GPUs with 16 GB VRAM. For gradient descent, we used Riemmanian Adam [4] with standard β values of 0.9 and 0.999 and an initial learning rate of 0.001. Number of dimensions (D) and number of layers (L) is empirically selected based on performance-memory trade-off. Figure 4.6 presents the memory-performance trade-off for different choices of parameters D and L. We observe that the $D = 8$ and $L = 8$ provides the best performance for the memory required. Hence, we chose them for the final implementation of our model. For non-linearity, we used the hyperbolic activation function, given in Eq. (4.1). The sparsity in the model variables is handled using the torch-sparse library[6]. While this library and other similar ones handle the operational sparsity of the graphs, previous GNN-based approaches need to locally convert the sparse tensors to the corresponding dense format for their layer operations. In TESH-GCN, the conversion is not required because **all operations in Sparse-HGCN are directly performed on the sparse tensor as it only considers the non-zero elements of the tensor.** Each convolution operation moves up one-hop in the nodes' neighborhood. Hence, the number of graph convolution layers should at least be the maximum shortest

[6]https://github.com/rusty1s/pytorch_sparse

path between any two nodes in the graph. For a dataset, this is empirically calculated by sampling nodes from the graph and calculating the maximum shortest path between them. For the datasets in our experiments, we used 8 layers ($L = 8$) to extract local neighborhoods in the early layers and metapath structures in the later layers. The main adjacency tensor can be split either over the number of semantic signals (D) or the number of edge types (K). We chose the latter because each adjacency tensor needed a separate GPU and it was more efficient and convenient to control the training process, given that the number of edge types is lesser than the number of semantic signals in our experiments. Algorithm 4 provides the pseudocode for training the TESH-GCN model. The methodology for extracting the metapaths with their corresponding weightage in the final link prediction is presented in Algorithm 5.

4.4 Experimental Setup

In this section, we describe our experimental setup and investigate the following research questions (**RQs**):

1. **RQ1:** Does TESH-GCN perform better than the state-of-the-art approaches for the task of link prediction?
2. **RQ2:** What is the contribution of TESH-GCN's individual components to the overall performance?
3. **RQ3:** How does TESH-GCN compare against previous approaches in time and space complexity?
4. **RQ4:** How robust is TESH-GCN against noise in the graph and its corresponding text?
5. **RQ5:** Can we comprehend the results of TESH-GCN?

4.4.1 Datasets Used

For the datasets, we select the following widely used publicly available network benchmark datasets where the nodes contain certain semantic information in the form of text attributes. Also, the choice of the datasets is driven by the diversity of their hyperbolicity to test performance on different levels of latent hierarchy (lower hyperbolicity implies more latent hierarchy).

1. **Amazon** [57] is a heterogeneous e-commerce graph dataset that contains electronic products as nodes with title text connected by edges based on the purchase information.

Algorithm 4: TESH-GCN training

Data: Training data $(v_i, s_i, v_j, s_j, \hat{y_k}) \in E$;
Output: Predictor P_θ;

Initialize model parameters θ;
for *number of epochs; until convergence* **do**
  $l = 0$; # Initialize loss
  for $\{(v_i, s_i, v_j, s_j, \hat{y_k}) \in E\}$ **do**
    $t_i \leftarrow LM(s_i)$, $t_j \leftarrow LM(s_j)$;
    for $e_k \in E$ **do**
      # Stack D-repetitions of adjacency matrix
      $A_k = stack(E_k, D)$;
      $A_k[d, i, :] = t_i[d]$, $A_k[d, j, :] = t_j[d]$
      $x_0 = A_k$
      # Run through L graph convolution layers
      for $l : 1 \rightarrow L$ **do**
        $o_{p,l} = W^f \otimes_{c_l} x_{p,l-1} \oplus_{c_l} b_l \quad \forall x_{p,l-1} \neq 0$
        $a_{p,l} = exp^{c_l}\left(\frac{\alpha_p \log^{c_l}(o_{p,l})}{\sum_p \alpha_p \log^{c_l}(o_{p,l})}\right)$
        $h_{p,l} = \sigma_{c_l}(a_{p,l})$
      end
      $h_{k,L} = h_{p,l}$
    end
    # Attention over outputs
    $h_{k,L} = \frac{\alpha_k h_{k,L}}{\sum_k \alpha_k h_{k,L}}$
    $h_L = h_{1,L} \odot h_{2,L} \odot ... \odot h_{k,L}$
    $out(A) = h_L \odot t_i \odot t_j$
    # Predicted class probability
    $y_k = softmax(dense(out(A)))$
    $l = l + L(y_k, \hat{y_k})$ # Update loss
  end
  $\theta \leftarrow \theta - \nabla_\theta l$; # Update parameters
end
return P_θ

Algorithm 5: Metapath Explanations

Input: Input (v_i, s_i, v_j, s_j), Predictor P_θ;
Output: Metapath set M, Class prediction y_k;

Initialize metapath set $M = \phi$;
$t_i \leftarrow LM(s_i)$, $t_j \leftarrow LM(s_j)$;
for $e_k \in E$ **do**
  Initialize metapath for e_k, $M_k = \phi$;
  # stack D-repetitions of adjacency matrix
  $A_k = stack(E_k, D)$;
  $A_k[d, i, :] = t_i[d]$, $A_k[d, j, :] = t_j[d]$
  $x_0 = A_k$
  # Run through L graph convolution layers
  for $l : 1 \rightarrow L$ **do**
    $o_{p,l} = W^f \otimes_{c_l} x_{p,l-1} \oplus_{c_l} b_l \quad \forall x_{p,l-1} \neq 0$
    $a_{p,l} = exp^{c_l}\left(\frac{\alpha_p \log^{c_l}(o_{p,l})}{\sum_p \alpha_p \log^{c_l}(o_{p,l})}\right)$
    $h_{p,l} = \sigma_{c_l}(a_{p,l})$
    $M_k = M_k \cup \arg\max_p h_{p,l}$
  end
  $h_{k,L} = h_{p,l}$
end
Attention over outputs
$h_{k,L} = \frac{\alpha_k h_{k,L}}{\sum_k \alpha_k h_{k,L}}$
Extracted metapath M_k with attention weight α_k
$M = M \cup (M_k, \alpha_k)$
$h_L = h_{1,L} \odot h_{2,L} \odot ... \odot h_{k,L}$
$out(A) = h_L \odot t_i \odot t_j$
Predicted class probability
$y_k = softmax(dense(out(A)))$
return M, y_k

Table 4.2: Dataset statistics including no. of nodes (V), edges (E), edge types (K), hyperbolicity (δ), and sparsity ratio (R).

Dataset	V	E	K	δ	R (%)
Amazon	368,871	6,471,233	2	2	99.99
DBLP	37,791	170,794	3	4	99.99
Twitter	81,306	1,768,149	1	1	99.97
Cora	2,708	5,429	1	11	99.92
MovieLens	10,010	1,122,457	3	2	99.00

The edge types are `also_buy` (products bought together) and `also_view` (products viewed in the same user session).

2. **DBLP** [65] is a heterogeneous relational dataset that contains papers, authors, conferences, and terms from the DBLP bibliography website connected by three edge types: `paper-author`, `paper-conf` and `paper-term`. For the semantic information, we include the paper's titles, author's names, conference's names, and the terms' text.

3. **Twitter** [73] dataset is a user follower network graph with unidentifiable profile information given as node's features. The node features are pre-encoded to remove sensitive identifiable information.

4. **Cora** [105] is a citation graph that contains publications with title text and author information connected by citation edges.

5. **MovieLens** [56] dataset is a standard user-movie heterogeneous rating dataset with three edge types: `user-movie`, `user-user`, and `movie-genre`. We utilize the movie's title and genre's name as the textual information.

In the case of graph-based methods, we utilize the node features provided in the dataset as default, else we utilize fixed-semantic vectors from the pretrained LM [113]. More detailed dataset statistics such as the number of nodes, edges, edge types, along with hyperbolicity and sparsity are given in Table 4.2.

4.4.2 Baselines

The selection of our baseline models was driven by two key factors; the diversity of methods employed, and their suitability to the datasets used in our experimental setup. To this end, we compare the performance of the proposed model with the following state-of-the-art models in the following categories: text-based (1-3), graph-based (4-6), and hybrid text-graph (7-9) approaches.

1. **C-DSSM** [109] is an extension of DSSM [63] that utilizes convolution layers to encode character trigrams of documents for matching semantic features.

2. **BERT** [34] is a popular transformer based language model that pre-trains on large amount of text data and is fine-tuned on sequence classification task for efficient text matching.

3. **XLNet** [145] is an improvement over the BERT model which uses position invariant autoregressive training to pre-train the language model.

4. **GraphSage** [54] is one of the first approaches that aggregate the neighborhood information of a graph's node. It includes three aggregators mean, LSTM [58], and max pooling. For our baseline, we choose the best performing LSTM aggregator.

5. **GCN** [70] utilizes convolutional networks to aggregate neighborhood information.

6. **HGCN** [17] utilizes convolutional networks in the hyperbolic space that typically performs better than the Euclidean counterparts, especially, for datasets with low hyperbolicity (i.e., more latent hierarchy).

7. **TextGNN** [159] initializes node attributes with semantic embeddings to outperform previous approaches especially for the task of link prediction.

8. **TextGCN** [146] constructs a word-document graph based on TF-IDF scores and then applies graph convolution for feature detection towards link prediction between nodes.

9. **Graphormer** [148] adds manually constructed global features using spatial encoding, centrality encoding, and edge encoding to the node vector to aggregate the neighborhood in a transformer network architecture for graph-level prediction tasks.

4.4.3 RQ1: Performance on Link Prediction

To analyze the performance of TESH-GCN, we compare it against the state-of-the-art baselines using standard graph datasets on the task of link prediction. We input the node-pairs (v_i, v_j) with the corresponding text sequence (s_i, s_j) to the model and predict the probability that an edge type e_k connects them as $y_k = P_\theta(e_k|(v_i, v_j, s_i, s_j))$. We evaluate our model using 5-fold cross validation splits on the following standard performance metrics: Accuracy (ACC), Area under ROC curve (AUC), Precision (P), and F-score (F1). For our experimentation, we perform 5-fold cross validation with a training, validation and test split of 8:1:1 on the edges of the datasets. Table 4.3 provides the number of samples and sparsity of each split in the dataset. The results on the test set are presented in Table 4.4.

Table 4.3: Splits of the dataset for the link prediction experiment (RQ1). N is the number of samples in each split and R(%) provides the sparsity ratio of the split.

Dataset	**Training**		**Validation**		**Test**	
	N	**R(%)**	**N**	**R(%)**	**N**	**R(%)**
Amazon	5,176,986	99.99	647,123	99.99	647,124	99.99
DBLP	1,36,635	99.99	17,079	99.99	17,080	99.99
Twitter	1,414,519	99.97	176,815	99.99	176,815	99.99
Cora	4,343	99.94	543	99.99	543	99.99
MovieLens	897,966	99.10	112,245	99.88	112,246	99.88

Table 4.4: Performance comparison of our proposed model against several state-of-the-art baseline methods across diverse datasets on the task of link prediction. Metrics such as Accuracy (ACC), Area under ROC (AUC), Precision (P), and F-scores (F1) are used for evaluation. The rows corresponding to w/o Text, w/o Hyperbolic, w/o Residual, and CE Loss represent the performance of TESH-GCN without the text information, hyperbolic transformation, residual connections, and with standard cross entropy loss (instead of multi-step loss), respectively. The best and second best results are highlighted in bold and underline, respectively. The improvement of TESH-GCN is statistically significant over the best performing baseline with a p-value threshold of 0.01.

	Datasets	**Amazon**				**DBLP**				**Twitter**				**Cora**				**MovieLens**			
	Models	**ACC**	**AUC**	**P**	**F1**	**ACC**	**AUC**	**P**	**F1**	**ACC**	**AUC**	**P**	**F1**	**ACC**	**AUC**	**P**	**F1**	**ACC**	**AUC**	**P**	**F1**
Text	C-DSSM	.675	.681	.677	.674	.518	.522	.519	.513	.593	.595	.588	.586	.693	.697	.696	.693	.664	.660	.658	.660
	BERT	.787	.793	.797	.784	.604	.605	.605	.603	.667	.664	.630	.641	.757	.763	.758	.751	.760	.764	.757	.752
	XLNet	.788	.793	.797	.785	.602	.602	.610	.604	.626	.626	.651	.654	.761	.768	.762	.758	.750	.758	.766	.754
Graph	GraphSage	.677	.680	.679	.673	.520	.525	.519	.518	.591	.592	.588	.585	.809	.813	.813	.805	.660	.659	.662	.656
	GCN	.678	.679	.679	.674	.412	.412	.413	.401	.564	.566	.553	.545	.813	.817	.818	.814	.652	.652	.649	.650
	HGCN	.710	.715	.712	.703	.547	.548	.544	.533	.608	.605	.580	.598	**.929**	**.934**	**.931**	**.923**	.685	.697	.687	.677
Hybrid	TextGNN	.742	.742	.744	.732	.567	.573	.573	.562	..636	.636	.628	.621	.843	.848	.848	.840	.723	.724	.719	.712
	TextGCN	.817	.824	.818	.809	.624	.626	.625	.616	.671	.670	.660	.669	.862	.864	.870	.856	.789	.790	.783	.780
	Graphormer	.804	.808	.806	.804	.617	.619	.621	.612	.692	.693	.669	.666	.849	.851	.858	.849	.780	.780	.779	.771
Ours	TESH-GCN	**.829**	**.836**	**.837**	**.836**	**.636**	**.640**	**.644**	**.640**	**.709**	**.710**	**.671**	**.670**	.909	.901	.902	.908	**.806**	**.814**	**.801**	**.801**
	w/o Text	.784	.784	.784	.784	.599	.605	.612	.599	.645	.648	.648	.622	.854	.858	.842	.824	.759	.753	.756	.748
	w/o Hyperbolic	.677	.672	.678	.678	.522	.526	.531	.516	.577	.572	.554	.585	.787	.789	.781	.757	.655	.652	.651	.660
	w/o Residual	.826	.825	.829	.829	.629	.632	.640	.632	.699	.705	.662	.658	.937	.942	.929	.913	.796	.799	.788	.795
	CE Loss	.827	.830	.833	.832	.635	.635	.642	.639	.706	.707	.668	.665	.931	.939	.927	.916	.800	.805	.798	.795

From the experimental results, we observe that TESH-GCN is able to outperform the previous approaches by a significant margin on different evaluation metrics. Additionally, we notice that the performance improvement of hyperbolic models (HGCN and TESH-GCN) is more on datasets with lower hyperbolicity (higher latent hierarchy). This shows that hyperbolic space is better at extracting hierarchical features from the graph structures. Furthermore, we see that the performance decreases a little without the residual network. However, it does not justify the additional parameters but it adds robustness against noisy graph and text (evaluation in Section 4.4.6), so we use this variant in our final model. Another point of note is that text-based frameworks are better than graph approaches in datasets with good semantic information such as Amazon, whereas, graph-based approaches are better on well-connected graphs such as Cora. However, TESH-GCN is able to maintain good performance in both the scenarios, *demonstrating its ability to capture both semantic and structural information from the dataset.*

4.4.4 RQ2: Ablation Study

In this section, we study the importance of different components and their contribution to the overall performance of our model. The different components we analyze in our ablation study are: (i) the semantic text signal, (ii) the hyperbolic transformations, (iii) the residual network, and (iv) the multi-step loss. The ablation study is conducted on the same datasets

by calculating the evaluation metrics after freezing the parameters of the component of interest in the model. The results of the study are presented in Table 4.4.

The results show that the text signal contributes to 7% performance gain in our model, implying the importance of utilizing the nodes' semantic information in aggregating features from the adjacency tensors. The hyperbolic transformations lead to a 18% increase in TESH-GCN's performance, demonstrating the importance of hierarchical features in extracting information from graphs. This also provides additional evidence of the latent hierarchy in the graph networks. Furthermore, removing the residual network shows a decrease of 1% in our model's performance which shows that text signals capture the semantic signal in the graph convolution layers and the residual network works only towards increasing the robustness in the final link prediction task. In addition to this, we notice that replacing multi-step loss with a standard cross entropy loss (with non-existence of links added as another class) leads to a 2% reduction in performance. This provides evidence for the advantages of conditioning link classification on link prediction (as in multi-step loss) compared to a standard multi-class loss function.

4.4.5 RQ3: Complexity Analysis

One of the major contributions of TESH-GCN is its ability to efficiently handle sparse adjacency tensors in its graph convolution operations. To compare its performance to previous graph-based and hybrid approaches, we analyze the space and time complexity of our models and the baselines. The space complexity is studied through the number of parameters and time complexity is reported using the training and inference times of the models. We compare the space and time complexity of our models using large graphs of different sparsity ratios (R) (by varying the number of edges/links on a graph with 10^4 nodes). The different sparsity ratios considered in the evaluation are $1 - 10^{-r}\ \forall r \in [\![0, 5]\!]$. Figure 4.7 and Table 4.5 shows the comparison of different GCN based models' training time on varying sparsity ratios and inference times on different datasets, respectively. Table 4.6 presents the number of parameters and space complexity of the different baselines in comparison to TESH-GCN.

From the time complexity analysis, we notice that TESH-GCN consistently takes much

Table 4.5: Inference times (in milliseconds) of our model and various GCN-based baseline methods.

Models	**Amazon**	**DBLP**	**Twitter**	**Cora**	**MovieLens**
GCN	719	723	728	735	744
HGCN	745	757	758	763	774
TextGNN	1350	1368	1375	1394	1395
TextGCN	1392	1416	1417	1431	1437
Graphormer	1423	1430	1441	1442	1458
TESH-GCN	787	794	803	817	822

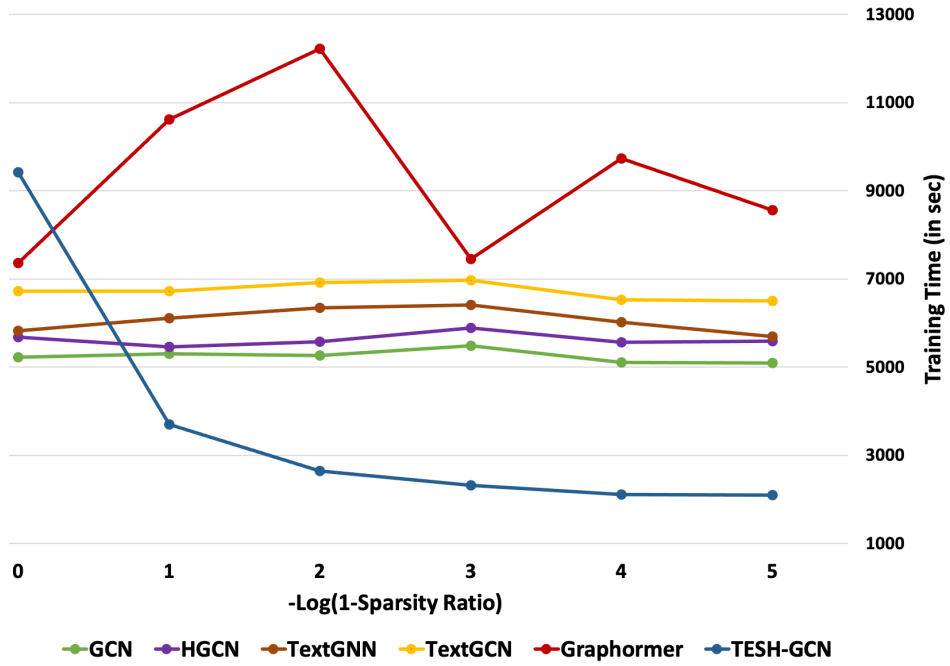


Figure 4.7: Comparison of training time (in seconds) of different GCN-based baseline methods on datasets with varying sparsity ratios (R).

less training time than the other GCN-based and hybrid approaches in high sparsity graphs. This shows that the current GCN-based approaches do not handle the sparsity of the adjacency tensor. However, the overhead of specialized graph convolution layer in TESH-GCN leads to a poor time complexity for cases with high graph density ($R < 0.9$). From the comparison of inference times, given in Table 4.5, we notice that TESH-GCN's inference time is comparable to the graph-based baselines and significantly lesser than hybrid baselines. Figure 4.8 provides the effect of sparsity on the inference time of our model and the baselines. We note that TESH-GCN is able to outperform other hybrid graph-text baselines and needs similar inference time as the baselines that only consider the local neighborhood of its nodes. TESH-GCN is faster for high sparsity graphs but the overhead of specialized graph convolutions takes more time than other baselines on high density graphs.

The space complexity analysis clearly shows that TESH-GCN uses much lesser number of model parameters than baselines with comparable performance. Also, the complexity shows the dependence of text-based approaches on only the textual sequence length, whereas, the graph based are dependent on the number of nodes. However, TESH-GCN is able to reduce the space complexity by a factor of the sparsity ratio and only consider informative non-zero features from the adjacency tensors, leading to a decrease in the number of trainable

[7]Note that, in the case of GNN-based networks, the basic formulations use sparse graph representations, which makes their complexity linear in the number of edges. However, in practice, GPU machines do not support sparse representations, and hence, GCNs need to be operated on dense adjacency matrices which leads to a time complexity of $\mathcal{O}(V^2)$.

Table 4.6: The number of non-trainable (in millions) and trainable (in thousands) parameters of all the comparison methods. We also report the space complexity in terms of the number of nodes (V), maximum text length (S), and sparsity measure $\left(N = \frac{1}{1-R} \approx 10^4\right)$[7].

Model	Non-Train (M)	Train (K)	Complexity
C-DSSM	0	38	$\mathcal{O}(S)$
BERT	110	1600	$\mathcal{O}(S^2)$
XLNet	110	1600	$\mathcal{O}(S^2)$
GraphSage	0	4800	$\mathcal{O}(V^2)$
GCN	0	4800	$\mathcal{O}(V^2)$
HGCN	0	9600	$\mathcal{O}(2V^2)$
TextGNN	110	6400	$\mathcal{O}(SV^2)$
TextGCN	110	6400	$\mathcal{O}(SV^2)$
Graphormer	100	7600	$\mathcal{O}(SV^2)$
TESH-GCN	110	78	$\mathcal{O}\left(\frac{2SV^2}{N}\right)$

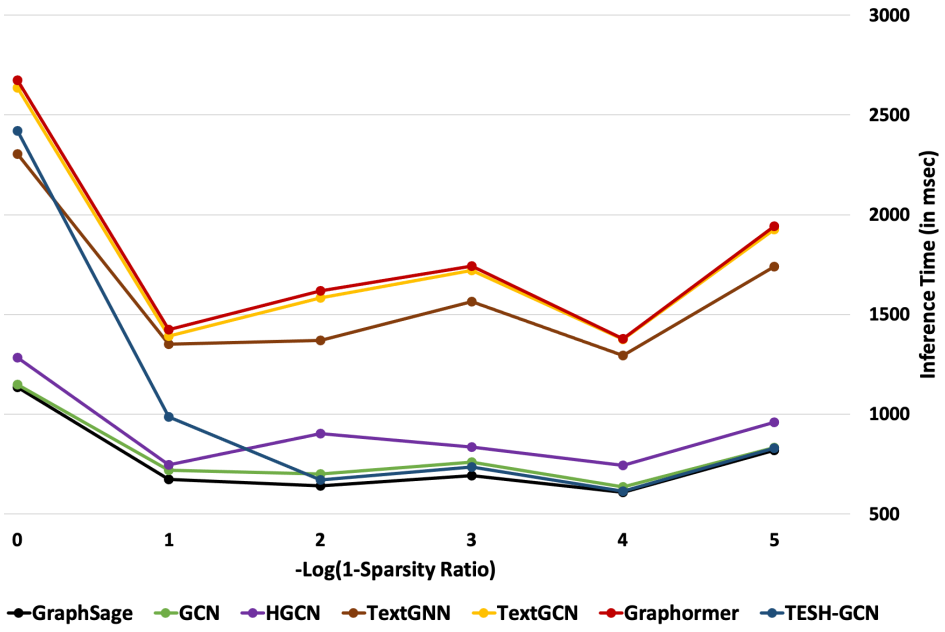
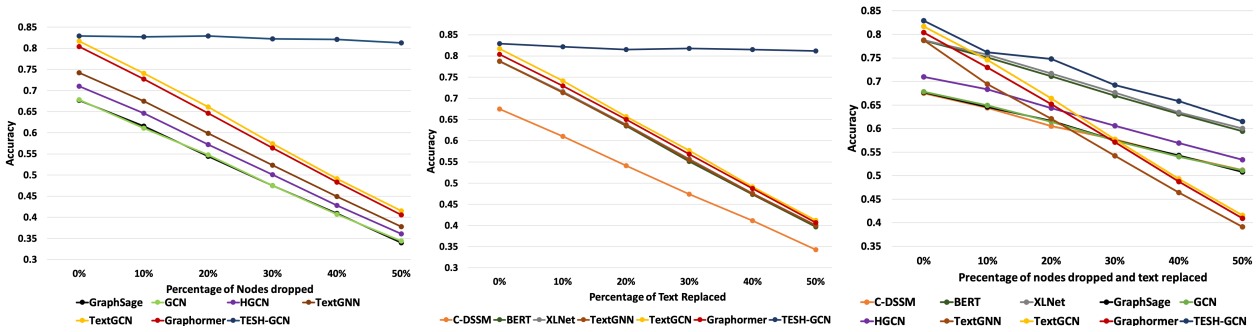


Figure 4.8: -log(1-R) vs Inference time (in milliseconds). Comparison of inference time of different baselines on a simulated dataset with 10,000 nodes and varying sparsity ratios (R).

parameters.



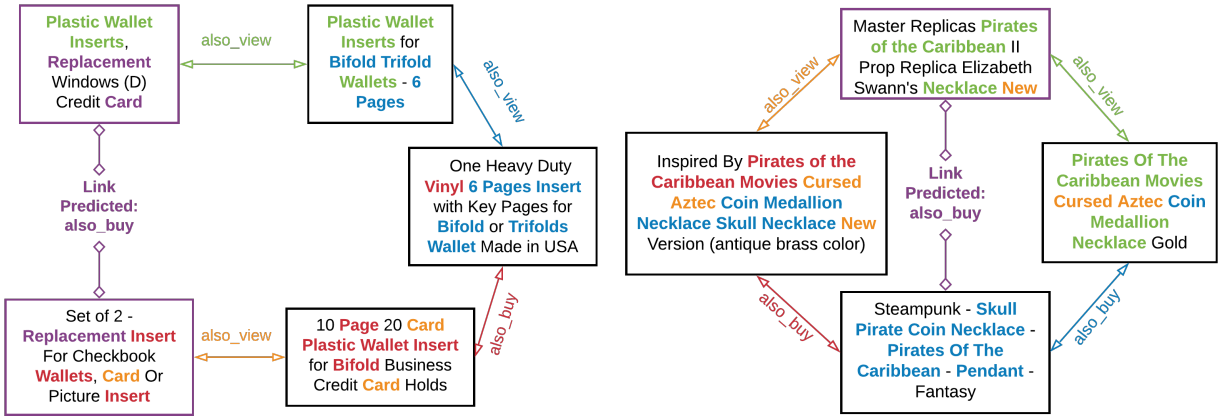
(a) Nodes dropped (%) vs Accuracy (b) Text replaced (%) vs Accuracy (c) Hybrid noise (%) vs Accuracy

Figure 4.9: Comparison of the effect of different noise-inducing methods on the accuracy of our model and the baselines. Noise is induced using (a) Node drop, (b) Text replacement, and (c) Hybrid noise (node drop and text replacement).

4.4.6 RQ4: Model Robustness

To test the robustness of our model, we introduce varying levels of noise into the Amazon graph by (i) *node drop*: dropping n% percentage of nodes, (ii) *text replacement*: replacing n% percentage of the text, and (iii) *hybrid noise*: dropping n% of nodes and replacing n% of text. We compare the performance of our model and the baselines across different values of $n = 10, 20, 30, 40$, and 50. The results for the robustness evaluation are given in Figure 4.9.

First, we highlight the main observations, that node drop and text replacement only affects graph-based and text-based approaches, respectively (and does not affect them vice versa). In the case of hybrid baselines, we still note a decrease in performance for both the noise variants. This implies that the text and graph features in the baselines do not complement each other. In the case of TESH-GCN, we note that both the noise variants do not cause any significant performance loss. This shows that the complementary nature of the semantic residual network and hyperbolic graph convolution network leads to an increased robustness against noise in either the text or graph. In the third scenario with hybrid noise, we see a reduction of $\approx 25\%$ performance in text-based and graph-based baselines and $\approx 50\%$ in hybrid baseline with a 50% noise. However, we notice that, although TESH-GCN is a hybrid model, we only observe a 25% performance loss with 50% noise, implying the effectiveness of text-graph correspondence in the scenario of hybrid noise as well. Thus, we conclude that TESH-GCN is robust against noise in either graph or text, but vulnerable, albeit less than other hybrid baselines, to a joint attack on both the graph and text.



(a) Aggregating information from long metapaths.

(b) Aggregating information from multiple metapaths.

Figure 4.10: Predictions showing TESH-GCN's metapath aggregation ability over both text and graphs. The local neighborhood and long metapath information is extracted in the early and later graph convolution layers, respectively. The textual information is extracted using attention over the semantic residual network. The colors assigned to the text match the color of the link through which the semantic information was passed to the ultimate nodes for message aggregation and subsequently link prediction. The samples are taken from the heterogeneous Amazon dataset.

4.4.7 RQ5: Model Explainability

Model comprehension is a critical part of our architecture as it helps us form a better understanding of the results and explain the model's final output. To understand TESH-GCN's link prediction, we look at the different metapaths that connect the input nodes as well as the text in the metapaths' nodes that receive the most attention (α_k). For this, we follow the graph convolution and attention pooling operations through the layers in the network and extract the most critical metapaths chosen by the model to arrive at the prediction. The methodology for extracting the metapaths with their corresponding weightage in the final link prediction is presented in Algorithm 5. Figure 4.10 depicts some metapaths extracted from the Amazon dataset. In Figures 4.10(a) and 4.10(b), we note that TESH-GCN aggregates information from multiple long (4-hop) metapaths between the input nodes for prediction. Additionally, we see tokens in the node's text being emphasized (having higher attention weight) based on the edges through which they propagate their semantic information, e.g., in Figure 4.10(b), we observe that key tokens: `Pirates of the Caribbean` and `Necklace` propagate the semantic information to match with additional relevant tokens such as `Cursed Aztec, Medallion, Pendant` and `coin` to establish the edge `also_buy` between the input nodes. Thus, *we observe the role of different metapaths as well as semantic information in the message propagation towards the downstream task of link prediction.*

4.5 Summary

In this chapter, we introduced Text Enriched Sparse Hyperbolic Graph Convolution Network (TESH-GCN), a hybrid graph and text based model for link prediction. TESH-GCN utilizes semantic signals from nodes to aggregate intra-node and inter-node information from the sparse adjacency tensor using a reformulated hyperbolic graph convolution layer. We show the effectiveness of our model against the state-of-the-art baselines on diverse datasets for the task of link prediction and evaluate the contribution of its different components to the overall performance. Additionally, we demonstrate the optimized memory and faster processing time of our model through space and time complexity analysis, respectively. Furthermore, we also show TESH-GCN's robustness against noisy graphs and text and provide a mechanism for explaining the results produced by the model.

Chapter 5

A Meta Learning Model for Scalable Hyperbolic Graph Neural Networks

Hyperbolic neural networks (HNNs) have outperformed their Euclidean counterparts in several domains that involve hierarchical datasets including recommender systems, biology, and knowledge graphs. However, current research in the domain is limited due to HNNs' lack of inductive bias mechanisms that could help them generalize well over unseen tasks or enable scalable learning over large datasets. In this chapter, we aim to alleviate these issues by learning generalizable inductive biases from the nodes' local subgraph and transfer them for faster learning over new subgraphs with a disjoint set of nodes, edges and labels in a few-shot setting. We theoretically justify that HNNs predominantly rely on local neighborhoods for label prediction evidence of a target node or edge, and hence, we learn the model parameters on local graph partitions, instead of the earlier approach that considers the entire graph together. We introduce a novel Hyperbolic GRAph Meta Learner (H-GRAM) that learns transferable information from a set of support local subgraphs, in the form of hyperbolic meta gradients and label hyperbolic protonets, to enable faster learning over a query set of new tasks dealing with disjoint subgraphs. Furthermore, we show that an extension of our meta-learning framework also solves the limitation of scalability in hyperbolic neural networks faced by earlier approaches. Our comparative analysis shows that H-GRAM effectively learns and transfers information in multiple challenging few-shot settings compared to other state-of-the-art baselines. Additionally, we demonstrate that, unlike standard HNNs, our model is able to efficiently scale over large standard graph datasets and improve performance over its Euclidean counterparts. Furthermore, we also evaluate the utility of various meta information components through an ablation study and analyze the performance of our algorithm over challenging few-shot learning scenarios.

5.1 Introduction

Graphs are extensively being used in a variety of applications, such as image processing, natural language processing, chemistry and bio informatics. Modern graph datasets range from hundreds of thousands to over a billion nodes. Thus, research in graph analysis has

steadily moved towards larger and more complex graphs, e.g., the nodes and edges in the traditional Cora dataset [105] are in the order of 10^3, whereas, the more recent ogbn datasets [60] are in the order of 10^6. Standard Graph Neural Networks (GNNs), generally modeled in the Euclidean space, that achieve promising results on several of the above domains were originally built to use the entire graph [33]. But as the graphs get larger, they run into scalability issues. These are circumvented in modern GNNs by focusing on an immediate (k-hop) neighborhood around a node, and learning an aggregated node or graph level representation of the same [20, 41].

Hyperbolic Graph Neural Networks (HNNs) have shown to outperform their Euclidean counterparts by taking advantage of the inherent hierarchy present in many modern datasets [161, 162]. Unlike a standard GNN, an HNN learns node representations based on an "anchor" or a "root" node for the entire graph, and operations needed to learn these embeddings are a function of this root node. Specifically, HNN formulations [45] depend on the global origin (root node) for several transformation operations (Möbius addition, Möbius multiplication and others), and hence focusing on subgraph structures to learn representations becomes meaningless when their relationship to the root node is not considered. Thus, state-of-the-art HNNs such as HGCN [17], HAT [50], and HypE [26] require access to the entire dataset to learn representations, and hence only scale to experimental sized datasets (with $\approx 10^3$ nodes). Despite this major drawback, HNNs have shown impressive performance on several research domains including recommendation systems [27, 117], e-commerce [28], natural language processing [35], and knowledge graphs [18, 26, 30]. It is thus imperative that one develops methods to scale HNNs to larger datasets, so as to realize their full potential. To this end, we introduce a novel method, Hyperbolic GRAph Meta Learner (H-GRAM), that utilizes meta-learning to learn information from local subgraphs for HNNs and transfer it for faster learning on a disjoint set of nodes, edges and labels contained in the larger graph. As a consequence of meta-learning, H-GRAM also achieves several desirable benefits that extends HNNs' applicability including the ability to transfer information on new graphs (inductive learning), elimination of over-smoothing, and few-shot learning. We experimentally show that H-GRAM can scale to graphs of size $\mathcal{O}(10^6)$, which is $\mathcal{O}(10^3)$ times larger than previous state-of-the-art HNNs. To the best of our knowledge, there do not exist other methods that can scale HNNs to datasets of this size.

Recent research has shown that both node-level and edge-level tasks only depend on the local neighborhood for evidence of prediction [62, 158]. Inspired by the insights of such research, our model handles large graph datasets using their node-centric subgraph partitions, where each subgraph consists of a root node and the k-hop neighborhood around it. In H-GRAM, the HNN formulations establish the root node as the local origin to encode the subgraph. We theoretically show that due to the locality of tangent space transformations in HNNs (to be made clear in Section 5.3), the evidence for a node's prediction can predominantly be found in the immediate neighborhood, and hence, the subgraph encodings do not lose a significant amount of feature information despite not having access to a "global" root node. However, due to the node-centric graph partitioning, the subgraphs are non-exhaustive, i.e.,

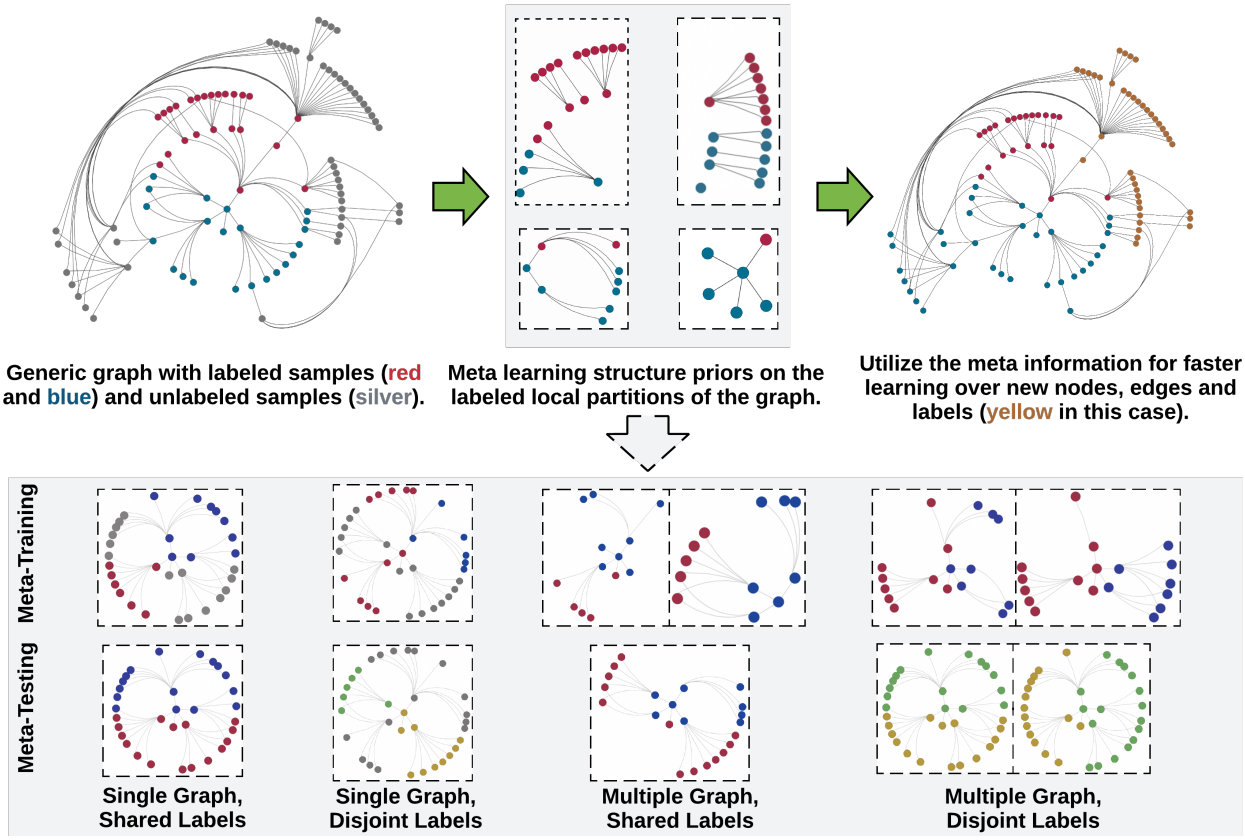


Figure 5.1: Meta-learning on hyperbolic neural networks. The procedure consists of two phases - (i) meta-training to update the parameters of the HNNs and learn inductive biases (meta gradients and label protonets), and (ii) meta-testing that initializes the HNNs with the inductive biases for faster learning over new graphs with a disjoint set of nodes, edges, or labels.

they do not contain all the nodes, edges and labels, as previously assumed by HNNs. Thus, to overcome the issue of non-exhaustive subgraphs, we formulate four meta-learning problems (illustrated in Figure 5.1) that learn inductive biases on a support set and transfers it for faster learning on a query set with disjoint nodes, edges or labels.

We consider four distinct problems for our meta-learning setup to generalize over all the possible outcomes of the graph partition; (i) Single Graph and Shared Labels, (ii) Single Graph and Disjoint Labels, (iii) Multiple Graphs and Shared Labels, and (iv) Multiple Graphs and Disjoint Labels. The problem setups are explained in further detail in Section 5.3.2. The subgraphs are combined together into task batches and divided into support and query set, based on the meta-learning setup. Given that the labels are not exhaustive (disjoint labels case), we cannot use categorical label encoding for training our models, and hence, we adopt continuous label protonets [112] instead. Label protonets construct class-specific representations, referred to as "label prototypes", by aggregating meta-training samples. These prototypes are then employed to classify meta-testing samples based on their similarity to the corresponding meta-training samples. This enables our model to handle new, non-exhaustive labels in an inductive manner.. Our model learns inductive biases from the support set in the meta-training phase, which consists of two steps; HNN update and meta update. HNN updates are regular stochastic gradient descent steps based on the loss of each support task. The updated HNN parameters are used to calculate the loss on query tasks and the gradients are accumulated into a meta-gradient for the meta update. In the meta-testing phase, the models are evaluated on the query tasks with parameters post the meta updates (using transferred meta gradients), as this snapshot of the model is the most adaptable for faster learning in a few-shot setting. Our main contributions are as follows:

1. We theoretically prove that HNNs rely on the nodes' local neighborhood for evidence in prediction, as well as, formulate HNNs to encode node-centric local subgraphs with root nodes as the local origin using the locality of tangent space transformations.
2. We develop Hyperbolic GRAph Meta Learner (H-GRAM), a novel method that learns meta information (as meta gradients and label protonets) from local subgraphs and generalize it to new graphs with a disjoint set of nodes, edges and labels. Our experiments show that H-GRAM can be used to generalize information from subgraph partitions of large datasets, thus, enabling scalability in hyperbolic models.
3. Our analysis on a diverse set of datasets demonstrates that our meta-learning setup also solves several challenges in HNNs including inductive learning, elimination of over-smoothing and few-shot learning in several challenging scenarios.

The rest of the chapter is organized as follows: Section 5.2 discusses the background work related to our work. Section 5.3 and 5.5 details our proposed model and experimental setup, respectively. Section 5.6 presents the experimental results of the proposed model and addresses several research questions. Finally, Section 5.8 summarizes the chapter.

5.2 Related Work

In this section, we review the relevant research in the areas of hyperbolic neural networks and meta learning.

Hyperbolic Neural Networks: Due to their ability to efficiently encode tree-like structures, hyperbolic space has been a significant development in the modeling of hierarchical graph datasets [161, 162]. Among its different isometric interpretations, the Poincaré ball model is the most popular one and has been widely applied in several HNN formulations of Euclidean networks including the recurrent (HGRU [45]), convolution (HGCN [17]), and attention layers (HAT [50]). As a result of their performance gains on hierarchical graphs, the formulations have also been extended to applications in knowledge graphs for efficiently encoding the hierarchical relations in different tasks such as representation learning (MuRP [3], AttH [18]) and logical reasoning (HypE [26]). However, the above approaches have been designed for experimental datasets with a relatively small number of nodes (in the order of 10^3), and do not scale to real-world datasets. Hence, we have designed H-GRAM as a meta-learning algorithm to translate the performance gains of HNNs to large graph datasets in a scalable manner.

Graph Meta-learning: Few-shot meta-learning transfers knowledge from prior tasks for faster learning over new tasks with few labels. Due to their wide applicability, they have been adopted in several domains including computer vision [46, 125], natural language processing [72] and, more recently, graphs [62, 153]. One of early approaches in graph meta-learning is Gated Propagation Networks [78] which learns to propagate information between different label prototypes to improve the information available while learning new related labels. Subsequent developments such MetaR [21], Meta-NA [157] and G-Matching [153] relied on metric-based meta learning algorithms for relational graphs, network alignment and generic graphs, respectively. These approaches show impressive performance on few-shot learning, but are only defined for single graphs. G-Meta [62] extends the metric-based techniques to handle multiple graphs with disjoint labels. However, the method processes information from local subgraphs in a Euclidean GNN, and thus, is not as capable as hyperbolic networks in encoding tree-like structures. Thus, we model H-GRAM to effectively encode hierarchical information from local subgraphs and transfer it to new subgraphs with disjoint nodes, edges, and labels.

5.3 Proposed Model

In this section, we discuss the preliminaries of hyperbolic operations, define the problem setup for different meta-learning scenarios and describe our proposed model, H-GRAM, illustrated in Figure 5.2. For better clarity, we explain the problem setup for node classification and use HGCN as the exemplar HNN model. However, the provided setup can easily be extended to

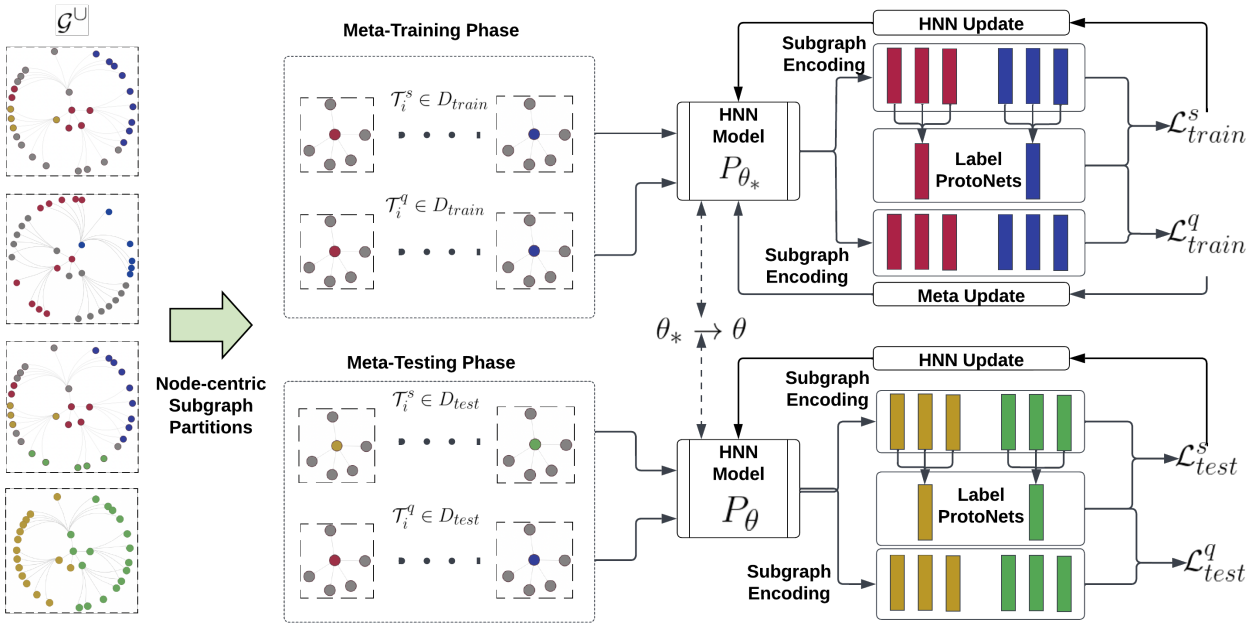


Figure 5.2: An Overview of the proposed H-GRAM meta-learning framework. Here, the input graphs $\mathcal{G}^{\cup}$ are first partitioned into node-centric subgraph partitions. We theoretically show that encoding these subgraph neighborhoods is equivalent to encoding the entire graph in the context of node classification and link prediction tasks. H-GRAM then uses an HGCN encoder to produce subgraph encodings, which are further utilized to get label prototypes. Using the HGCN gradient updates and label prototypes, the HNN model's parameters P_{θ^*} is updated through a series of weight updates and meta updates for η meta-training steps.The parameters are then transferred to the meta-testing stage $P_{\theta^* \rightarrow \theta}$ and further trained on $\mathcal{D}_{test}^s$ and evaluated on $\mathcal{D}_{test}^q$.

link prediction or to other HNN models, which we have evaluated in our experiments.

5.3.1 Preliminaries

In this section, we discuss the hyperbolic operations used in HNN formulations and set up the meta-learning problem.

Hyperbolic operations: The hyperbolic gyrovector operations required for training neural networks, in a Poincaré ball with curvature c, are defined by Möbius addition ($\oplus_c$), Möbius subtraction ($\ominus_c$), exponential map ($\exp_x^c$), logarithmic map ($\log_x^c$), Möbius scalar product ($\odot_c$), and Möbius matrix-vector product ($\otimes_c$).

$$
\begin{aligned}
g_x^{\mathbb{H}} &:= \lambda_x^2 \; g^{\mathbb{E}} \quad \text{where } \lambda_x := \frac{2}{1-\|x\|^2} \\
x \oplus_c y &:= \frac{(1+2c\langle x,y\rangle + c\|y\|^2)\,x + (1-c\|x\|^2)\,y}{1+2c\langle x,y\rangle + c^2\|x\|^2\|y\|^2} \\
x \ominus_c y &:= x \oplus_c -y \\
\exp_x^c(v) &:= x \oplus_c \left(\tanh\left(\sqrt{c}\frac{\lambda_x^c\|v\|}{2}\right)\frac{v}{\sqrt{c}\|v\|}\right) \\
\log_x^c(y) &:= \frac{2}{\sqrt{c}\lambda_x^c}\tanh^{-1}\left(\sqrt{c}\|-x\oplus_c y\|\right)\frac{-x\oplus_c y}{\|-x\oplus_c y\|} \\
r \odot_c x &:= \exp_0^c(r log_0^c(x)),\ \forall r \in \mathbb{R}, x \in \mathbb{H}^d \\
M \otimes_c x &:= \frac{1}{\sqrt{c}}\tanh\left(\frac{\|Mx\|}{\|x\|}tanh^{-1}\left(\sqrt{c}\|x\|\right)\right)\frac{Mx}{\|Mx\|}
\end{aligned}
$$

Here, $:=$ denotes assignment operation for Möbius operations. The operands $x, y \in \mathbb{H}^d$ are hyperbolic gyrovectors. $g^{\mathbb{E}} = \mathbf{I}_n$ 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[45].

Meta-Learning setup: In meta-learning, the dataset consists of multiple tasks $\mathcal{T}_i \in \mathcal{D}$. The dataset is divided into training ($\mathcal{D}_{train}$), test ($\mathcal{D}_{test}$), and validation ($\mathcal{D}_{val}$) sets, and each task $\mathcal{T}_i$ is divided into a support set ($\mathcal{T}_i^s$) and query set ($\mathcal{T}_i^q$) with corresponding labels Y_s and Y_q, respectively. The size of query label set is given by $N = |Y_q|$ and the size of support set in meta-testing is given by $K = |\mathcal{T}_i^s \in \mathcal{D}_{test}|$. This particular setup is also known as the N-ways K-shot learning problem. In the meta-training phase of MAML [43], the model trains on tasks $\mathcal{T}_i^s \in D_{train}$ and is evaluated on $\mathcal{T}_i^q \in \mathcal{D}_{train}$ to learn the meta-information. For few-shot evaluation, the model is trained on $\mathcal{T}_i^s \in \mathcal{D}_{test}$ and evaluated on $\mathcal{T}_i^q \in \mathcal{D}_{test}$, where $|\mathcal{T}_i^s| \ll |\mathcal{T}_i^q|$. The goal is to learn the initial parameters θ_* from $\mathcal{D}_{train}$ such that they can quickly transition to the parameters θ for new tasks in $\mathcal{D}_{test}$. The hyper-parameters of the meta-learning setup are tuned using $\mathcal{D}_{val}$.

5.3.2 Problem Setup

Our problem consists of a group of tasks $\mathcal{T}_i \in \mathcal{T}$ which are divided into a support set $\mathcal{T}^s$ and query set $\mathcal{T}^q$, where $\mathcal{T}^s \cap \mathcal{T}^q = \phi$. Furthermore, each task $\mathcal{T}_i$ is a batch of node-centric subgraphs S_u with a corresponding label Y_u (class of root node in node classification or root link in link prediction). The subgraphs S_u could be the partitions derived from a single graph or multiple graphs, both denoted by $\mathcal{G}^{\cup} = \mathcal{G}^s \cup \mathcal{G}^q$. We also define $Y_s = \{Y_u \in \mathcal{T}^s\}$ and $Y_q = \{Y_u \in \mathcal{T}^q\}$ as the set of labels in the support and query set respectively. The primary goal of meta-learning is to learn a predictor using the support set, $P_{\theta_*}(Y_s|\mathcal{T}^s)$ such that the model can quickly learn a predictor $P_\theta(Y_s|\mathcal{T}^s)$ on the query set in a few-shot setting. Following literature in this area [62], the problem categories are defined as follows:

1. **Single graph, shared labels $\langle SG, SL \rangle$**: The objective is to learn the meta-learning model $P_{\theta_* \to \theta}$, where $Y_s = Y_q$ and $|\mathcal{G}^s| = |\mathcal{G}^q| = 1$. It should be noted that the tasks are based on subgraphs, so $|\mathcal{G}^s| = |\mathcal{G}^q| = 1 \implies |\mathcal{T}^s| = |T^q| = 1$. Also, this problem setup is identical to the standard node classification task considering $\mathcal{T}_i^q \in D_{test}$ to be the test set.

2. **Single graph, disjoint labels $\langle SG, DL \rangle$**: This problem operates on the same graph in the support and query set, however unlike the previous one, the label sets are disjoint. The goal is to learn the model $P_{\theta_* \to \theta}$, where $|\mathcal{G}^s| = |\mathcal{G}^q| = 1$ and $Y_s \cap Y_q = \phi$.

3. **Multiple graphs, shared labels $\langle MG, SL \rangle$**: This problem setup varies in terms of the dataset it handles, i.e., the dataset can contain multiple graphs instead of a single one. However, our method focuses on tasks which contain node-centric subgraphs, and hence, the model's aim is the same as problem 1. The aim is to learn the predictor model $P_{\theta_* \to \theta}$, where $Y_s = Y_q$ and $|\mathcal{G}^s|, |\mathcal{G}^q| > 1$.

4. **Multiple graphs, disjoint labels $\langle MG, DL \rangle$**: In this problem, the setup is similar to the previous one, but only with disjoint labels instead of shared ones, i.e., learn a predictor model $P_{\theta_* \to \theta}$, where $Y_s \cap Y_q = \phi$ and $|\mathcal{G}^s|, |\mathcal{G}^q| > 1$.

From the problem setups, we observe that, while they handle different dataset variants, the base HNN model operates on the (S_u, Y_u) pair. So, we utilize a hyperbolic subgraph encoder and prototypical labels to encode S_u and get a continuous version of Y_u for our meta-learning algorithm, respectively.

5.3.3 Hyperbolic Subgraph Encoder

In previous methods [54, 62, 151], authors have shown that nodes' local neighborhood provides some informative signals for the prediction task. While the theory is not trivially extensible, we use the local tangent space of Poincaré ball model to prove that the local

neighborhood policy holds better for HNN models. The reason being that, while message propagation is linear in Euclidean GNNs [158], it is exponential in HNNs. Hence, a node's influence, as given by Definition 5.1, outside its neighborhood decreases exponentially.

Definition 5.1. The influence of a hyperbolic node vector $x_v^{\mathbb{H}}$ on node $x_u^{\mathbb{H}}$ is defined by the influence score $\mathcal{I}_{uv} = exp_0^c\left(\left\|\frac{\partial \log_0^x(x_u^{\mathbb{H}})}{\partial \log_0^x(x_v^{\mathbb{H}})}\right\|\right)$.

Definition 5.2. The influence of a graph $\mathcal{G}$ with set of vertices $\mathcal{V}$ on a node $u \in \mathcal{V}$ is defined as $\mathcal{I}_{\mathcal{G}}(u) = exp_0^c\left(\left(\sum_{v\in\mathcal{V}} log_0^c\left(\mathcal{I}_{uv}\right)\right)\right)$.

Theorem 5.3. *For a set of paths P_{uv} between nodes u and v, let us define $D_{g\mu}^{p_i}$ as the geometric mean of degree of nodes in a path $p_i \in P_{uv}$, p_{uv} as the shortest path, and $\mathcal{I}_{uv}$ as the influence of node v on u. Also, let us say $D_{g\mu}^{min} = min\left\{D_{g\mu}^{p_i} \forall p_i \in P_{uv}\right\}$, then the relation $\mathcal{I}_{uv} \leq exp_u^c\left(K/\left(D_{g\mu}^{min}\right)^{\|p_{uv}\|}\right)$ (where K is a constant) holds for message propagation in HGCN models.*

Theorem 5.3 shows that the influence of a node decreases exponent-of-exponentially ($exp_u^c = \mathcal{O}(e^n)$) with increasing distance $\|p_{uv}\|$. Thus, we conclude that encoding the local neighborhood of a node is sufficient to encode its features for label prediction.

Definition 5.4. The information loss between encoding an entire graph $\mathcal{G}$ and a subgraph S_u with root node u is defined as $\delta_{\mathbb{H}}(\mathcal{G}, S_u) = exp_0^c\left(log_0^c\left(\mathcal{I}_{\mathcal{G}}(u)\right) - log_0^c\left(\mathcal{I}_{S_u}(u)\right)\right)$.

Theorem 5.5. *For a subgraph S_u of graph $\mathcal{G}$ centered at node u, let us define a node $v \in \mathcal{G}$ with maximum influence on u, i.e., $v = \arg\max_t(\{\mathcal{I}_u t, t \in \mathcal{N}(u) \setminus u\})$. For a set of paths P_{uv} between nodes u and v, let us define $D_{g\mu}^{p_i}$ as the geometric mean of degree of nodes in a path $p_i \in P_{uv}$, $\|p_{uv}\|$ is the shortest path length, and $D_{g\mu}^{min} = min\left\{D_{g\mu}^{p_i} \forall p_i \in P_{uv}\right\}$. Then, the information loss is bounded by $\delta_{\mathbb{H}}(\mathcal{G}, S_u) \leq exp_u^c\left(K/\left(D_{g\mu}^{min}\right)^{\|p_{uv}\|+1}\right)$ (where K is a constant).*

Theorem 5.5 shows that encoding the local subgraph is a $e^{\|p_{uv}\|}$ order approximation of encoding the entire graph, and thus, with high enough $\|p_{uv}\|$, the encodings are equivalent. Note that $\|p_{uv}\|$ is equivalent to the neighborhood size (k in k-hop) of the subgraph. This shows that encoding the local neighborhood of a node is sufficient to encode its features for label prediction. Theorem proofs are provided in Section 5.4.

The above theorems provide us with the theoretical justification for encoding local subgraphs into our meta-learning framework. Hence, we partition the input $\mathcal{G}^{\cup}$ into subgraphs $S_u = V \times E$ centered at root node u, with $|V|$ nodes, $|E|$ edges, a neighborhood size k, and the corresponding label Y_u. The subgraphs are processed through an k-layer HGCN network and the Einstein midpoint [122] of all the node encodings are taken as the overall encoding of the subgraph S_u. For a given subgraph $S_u = (A, X)$, where $A \in \mathbb{H}^{\|V\|\times\|V\|}$ is the local

adjacency matrix and $X \in \mathbb{H}^{\|V\| \times m}$ are the node feature vectors of m dimension, the encoding procedure given can be formalized as:

$$h_u = HGCN_{\theta^*}(A, X), \text{ where } h_u \in \mathbb{H}^{\|V\| \times d} \tag{5.1}$$

$$e_u = \frac{\sum_{i=1}^{\|V\|} \gamma_{iu} h_{iu}}{\sum_{i=1}^{\|V\|} \gamma_{iu}}, \text{ where } \gamma_{iu} = \frac{1}{\sqrt{1 - \|h_{iu}\|^2}} \tag{5.2}$$

where $HGCN(A, X) \in \mathbb{H}^{\|V\| \times d}$ is the output of k-layer HGCN with d output units and $e_u \in \mathbb{H}^d$ is the final subgraph encoding of S_u. γ_{iu} is the Lorentz factor of the hyperbolic vector h_{iu} that indicates its weightage towards the Einstein midpoint.

5.3.4 Label Prototypes

Label information is generally categorical in node classification tasks. However, this does not allow us to pass inductive biases from the support set to the query set. Hence, to circumvent this issue, we use prototypical networks [112] as our label encoding. Our approach constructs continuous label prototypes by using the mean of meta-training nodes' features that belong to the label. These prototypes are then employed to classify meta-testing samples based on their similarity to the corresponding meta-training label prototypes. This enables our model to handle new, non-exhaustive labels in an inductive manner, without the need for additional training data. The primary idea is to form a continuous label prototypes using the mean of nodes that belong to the label. To this end, the continuous label prototype of a label y_k is defined as c_k;

$$c_k = \frac{\sum_{Y_u = y_k} \gamma_i e_i}{\sum_{Y_u = y_k} \gamma_i}, \text{ where } \gamma_i = \frac{1}{\sqrt{1 - \|e_i\|^2}} \tag{5.3}$$

where $e_i \in \mathbb{H}^d$ is encoding of subgraphs S_u with labels $Y_u = y_k$. For each S_u with class y_k, we compute the class distribution vector as p_k and the subsequent loss for HNN updates $\mathcal{L}(p, y)$ as follows:

$$p_k = \frac{e^{(-d_{\mathbb{H}}^c(e_u, c_k))}}{\sum_k e^{(-d_{\mathbb{H}}^c(e_u, c_k))}}, \text{ where} \tag{5.4}$$

$$d_{\mathbb{H}}^c(e_u, c_k) = \frac{2}{\sqrt{c}} \tanh^{-1}\left(\sqrt{c}\| - e_u \oplus c_k\|\right) \tag{5.5}$$

$$\mathcal{L}(p, y) = \sum_j y_i \log p_j \tag{5.6}$$

where y_i is the one-hot encoding of the ground truth. The class distribution vector p_k is a softmax over the hyperbolic distance of subgraph encoding to the label prototypes, which indicates the probability that the subgraph belongs to the class y_k. The loss function $\mathcal{L}(p, y)$ is the cross-entropy loss between ground truth labels y and the class distribution vector p.

5.3.5 Meta-Learning

In the previous section, we learned a continuous label encoding that is able to capture inductive biases from the subgraph. In this section, we utilize the optimization-based MAML algorithm [43] to transfer the inductive biases from the support set to the query set. To this end, we sample a batch of tasks, where each task $\mathcal{T}_i = \{S_i, Y_i\}_{i=1}^{\|\mathcal{T}_i\|}$. In the meta-training phase, we first perform the HNN parameter using the Riemannian stochastic gradient descent (RSGD) [10] on support loss, i.e., for each $\mathcal{T}_i^s \in \mathcal{T}_{train}^s : \theta_j^* \leftarrow exp_{\theta_j^*}^c(-\alpha \nabla \mathcal{L}^s)$, where α is the learning rate of RSGD. Using the updated parameters θ_j^*, we record the evaluation results on the query set, i.e., loss on task $\mathcal{T}_i^q \in \mathcal{T}_{train}^q$ is $\mathcal{L}_i^q$. The above procedure is repeated η times post which $\mathcal{L}_i^q$ over the batch of tasks $\mathcal{T}_i^q \in \mathcal{T}_{train}^q$ is accumulated for the meta update $\theta^* \leftarrow exp_{\theta^*}^c(-\beta \nabla \sum_i \mathcal{L}_i^q)$. The above steps are repeated with the updated θ^* and a new batch of tasks till convergence. The final updated parameter set $\theta^* \rightarrow \theta$ is transferred to the meta-testing phase. In meta-testing, the tasks $\mathcal{T}_i^s \in \mathcal{T}_{test}^s$ are used for RSGD parameter updates, i.e., $\mathcal{T}_i^s \in \mathcal{T}_{test}^s : \theta_j \leftarrow exp_{\theta_j}^c(-\alpha \nabla \mathcal{L}^s)$ until convergence. The updated parameters θ are used for the final evaluation on $\mathcal{T}_i^q \in \mathcal{T}_{test}^q$.

5.4 Theorem Proofs

This section provides the theoretical proofs of the theorems presented in the chapter.

5.4.1 Proof of Theorem 5.3

Theorem. For a set of paths P_{uv} between nodes u and v, let us define $D_{g\mu}^{p_i}$ as the geometric mean of degree of nodes in a path $p_i \in P_{uv}$, p_{uv} as the shortest path, and $\mathcal{I}_{uv}$ as the influence of node v on u. Also, let us say define $D_{g\mu}^{min} = min\left\{D_{g\mu}^{p_i} \forall p_i \in P_{uv}\right\}$, then the relation $\mathcal{I}_{uv} \leq exp_u^c\left(K/\left(D_{g\mu}^{min}\right)^{\|p_{uv}\|}\right)$ (where K is a constant) holds for message propagation in HGCN models.

Proof. The aggregation in HGCN model is defined as:

$$x_u^{\mathbb{H}} = exp_{x_u^{\mathbb{H}}}^c\left(\frac{1}{D_u}\sum_{i \in \mathcal{N}(u)} w_{ui} log_{x_u^{\mathbb{H}}}^c\left(x_i^{\mathbb{H}}\right)\right)$$

where x_u, D_u and $\mathcal{N}(u)$ are the embedding, degree and neighborhood of the root node, respectively. Note that points in the local tangent space follow Euclidean algebra. For simplicity, let us define the Euclidean vector as $x_i = log_{x_u^{\mathbb{H}}}^c\left(x_i^{\mathbb{H}}\right)$. Simplifying the aggregation

function:

$$x_u^{\mathbb{H}} = exp_{x_u^{\mathbb{H}}}^c \left(\frac{1}{D_u} \sum_{i \in \mathcal{N}(u)} w_{ui} x_i \right)$$

Expanding the aggregation function to cover all possible paths from u to its connected nodes.

$$x_u^{\mathbb{H}} = exp_{x_u^{\mathbb{H}}}^c \left(\frac{1}{D_u} \sum_{i \in \mathcal{N}(u)} w_{ui} \frac{1}{D_i} \sum_{j \in \mathcal{N}(i)} w_{ij} ... \frac{1}{D_m} \sum_{n \in \mathcal{N}(m)} w_{mn} ... \frac{1}{D_o} \sum_{o \in \mathcal{N}(k)} w_{ko} x_o \right)$$

The influence of a node v on u is given by:

$$\mathcal{I}_{uv} = exp_0^c \left(\left\| \frac{\partial log_x^c \left(x_u^{\mathbb{H}} \right)}{\partial log_x^c \left(x_v^{\mathbb{H}} \right)} \right\| \right)$$

Simplifying tangent space vectors,

$$\mathcal{I}_{uv} = exp_0^c \left(\left\| \frac{\partial x_u}{\partial x_v} \right\| \right)$$

$$\left\| \frac{\partial x_u}{\partial x_v} \right\| = \left\| \frac{\partial}{\partial x_v} \left(\frac{1}{D_u} \sum_{i \in \mathcal{N}(u)} w_{ui} \frac{1}{D_i} \sum_{j \in \mathcal{N}(i)} w_{ij} ... \frac{1}{D_m} \sum_{n \in \mathcal{N}(m)} w_{mn} ... \frac{1}{D_o} \sum_{o \in \mathcal{N}(k)} w_{ko} x_o \right) \right\|$$

Given that the partial derivative is with respect to x_v, only paths between u and v will be non-zero, all other paths shall be zero, i.e.,

$$\left\| \frac{\partial x_u}{\partial x_v} \right\| = \left\| \frac{\partial}{\partial x_v} \left(\frac{1}{D_u} w_{up_i^1} \frac{1}{D_{p_i^1}} w_{p_i^1 p_j^1} ... \frac{1}{D_{p_k^1}} w_{p_k^1 v} x_v + ... + \frac{1}{D_u} w_{up_i^m} \frac{1}{D_{p_i^m}} w_{p_i^m p_j^m} ... \frac{1}{D_{p_k^m}} w_{p_k^m v} x_v \right) \right\|$$

where $(u, p_i^t, p_j^t, ..., p_k^t, v) \forall t \in [1, m]$ are the paths between node u and v. Aggregating the terms together and getting the constants out of the derivative,

$$\begin{aligned} \left\| \frac{\partial x_u}{\partial x_v} \right\| &= \left\| \frac{w_{up_i^1} w_{p_i^1 p_j^1} ... w_{p_k^1 v}}{D_u D_{p_i^1} ... D_{p_k^1}} + ... + \frac{w_{up_i^m} w_{p_i^m p_j^m} ... w_{p_k^m v}}{D_u D_{p_i^m} ... D_{p_k^m}} \right\| . \left\| \frac{\partial x_v}{\partial x_v} \right\| \\ &= \left\| \frac{w_{up_i^1} w_{p_i^1 p_j^1} ... w_{p_k^1 v}}{D_u D_{p_i^1} ... D_{p_k^1}} + ... + \frac{w_{up_i^m} w_{p_i^m p_j^m} ... w_{p_k^m v}}{D_u D_{p_i^m} ... D_{p_k^m}} \right\| \\ &\leq \left\| m * max \left(\frac{w_{up_i^1} w_{p_i^1 p_j^1} ... w_{p_k^1 v}}{D_u D_{p_i^1} ... D_{p_k^1}}, ..., \frac{w_{up_i^m} w_{p_i^m p_j^m} ... w_{p_k^m v}}{D_u D_{p_i^m} ... D_{p_k^m}} \right) \right\| \end{aligned}$$

Let us say,

$$t^* = \arg\max_t \left(\left\{ \frac{w_{up_i^t} w_{p_i^t p_j^t} ... w_{p_k^t v}}{D_u D_{p_i^1} ... D_{p_k^t}} \right\}_{t=1}^{t=m} \right) \tag{5.7}$$

Then,

$$\left\|\frac{\partial x_u}{\partial x_v}\right\| \leq \left\| m * \frac{w_{up_i^{t^*}} w_{p_i^{t^*} p_j^{t^*}} \ldots w_{p_k^{t^*} v}}{D_u D_{p_i^{t^*}} \ldots D_{p_k^{t^*}}} \right\|$$

Aggregating the constants and substituting the geometric mean, we get;

$$\left\|\frac{\partial x_u}{\partial x_v}\right\| \leq K \times \left(\frac{1}{\left(D_u D_{p_i^{t^*}} \ldots D_{p_k^{t^*}}\right)^{1/n_*}} \right)^{n_*} = \frac{K}{\left(D_{g\mu}^{t^*}\right)^{n_*}}$$

Substituting the variables with shortest paths and minimum degree,

$$\left\|\frac{\partial x_u}{\partial x_v}\right\| \leq \frac{K}{\left(D_{g\mu}^{t^*}\right)^{n_*}} \leq \frac{K}{\left(D_{g\mu}^{min}\right)^{\|p_{uv}\|}}$$

With transitive property and adding exponential map on both sides;

$$exp_u^c\left(\left\|\frac{\partial x_u}{\partial x_v}\right\|\right) \leq exp_u^c\left(K / \left(D_{g\mu}^{min}\right)^{\|p_{uv}\|}\right)$$
$$\mathcal{I}_{uv} \leq exp_u^c\left(K / \left(D_{g\mu}^{min}\right)^{\|p_{uv}\|}\right)$$

□

5.4.2 Proof of Theorem 5.5

Theorem. Given the subgraph S_u of graph $\mathcal{G}$ centered at node u, with the corresponding label Y_u, let us define a node $v \in \mathcal{G}$ with maximum influence on u, i.e., $v = \arg\max_t(\{\mathcal{I}_u t, t \in \mathcal{N}(u) \setminus u\})$. For a set of paths P_{uv} between nodes u and v, let us define $D_{g\mu}^{p_i}$ as the geometric mean of degree of nodes in a path $p_i \in P_{uv}$, $\|p_{uv}\|$ is the shortest path length, and $D_{g\mu}^{min} = min\left\{D_{g\mu}^{p_i} \forall p_i \in P_{uv}\right\}$. Then, the information loss between encoding the $\mathcal{G}$ and S_u is bounded by $\delta_{\mathbb{H}}(\mathcal{G}, S_u) \leq exp_u^c\left(K / \left(D_{g\mu}^{min}\right)^{\|p_{uv}\|+1}\right)$ (where K is a constant).

Proof. The information loss between encoding the entire graph $\mathcal{G}$ and node-centric local

subgraph S_u with root node u is given by;

$$\begin{aligned}
\delta_{\mathbb{H}}(\mathcal{G}, S_u) =& exp_0^c\left(\delta(\mathcal{G}, S_u)\right) \\
\delta(\mathcal{G}, S_u) =& log_0^c\left(\mathcal{I}_{\mathcal{G}}(u)\right) - log_0^c\left(\mathcal{I}_{S_u}(u)\right) \\
=& \left(\left\|\frac{\partial x_u}{\partial x_1}\right\| + ... + \left\|\frac{\partial x_u}{\partial x_n}\right\|\right) - \left(\left\|\frac{\partial x_u}{\partial x_{i_1}}\right\| + ... + \left\|\frac{\partial x_u}{\partial x_{i_m}}\right\|\right) \\
& \text{Delete the overlapping nodes in the paths,} \\
=& \left\|\frac{\partial x_u}{\partial x_{t_1}}\right\| + \left\|\frac{\partial x_u}{\partial x_{t_2}}\right\| + ... + \left\|\frac{\partial x_u}{\partial x_{t_{n-m}}}\right\| \\
& \text{Using Theorem 5.3,} \\
\leq& \frac{K_{t_1}}{\left(D_{g\mu}^{t_1}\right)^{\|p_{uv}^{t_1}\|}} + \frac{K_{t_2}}{\left(D_{g\mu}^{t_2}\right)^{\|p_{uv}^{t_2}\|}} + ... + \frac{K_{t_{n-m}}}{\left(D_{g\mu}^{t_{n-m}}\right)^{\|p_{uv}^{t_{n-m}}\|}} \\
\leq& (n-m) \times K_{min} / \left(D_{g\mu}^{min}\right)^{\|p_{uv}^{min}+1\|} \\
\leq& (n-m) \times K_{min} / \left(D_{g\mu}^{min}\right)^{\|p_{uv}+1\|} = K / \left(D_{g\mu}^{min}\right)^{\|p_{uv}+1\|} \\
\delta(\mathcal{G}, S_u) \leq& K / \left(D_{g\mu}^{min}\right)^{\|p_{uv}+1\|} \\
exp_0^c\left(\delta(\mathcal{G}, S_u)\right) \leq& exp_0^c\left(K / \left(D_{g\mu}^{min}\right)^{\|p_{uv}+1\|}\right) \\
\delta_{\mathbb{H}}(\mathcal{G}, S_u) \leq& exp_u^c\left(K / \left(D_{g\mu}^{min}\right)^{\|p_{uv}\|+1}\right)
\end{aligned}$$

□

5.4.3 Implementation Details

H-GRAM is primarily implemented in Pytorch [100], with geoopt [71] and GraphZoo [128] as support libraries for hyperbolic formulations. Our experiments are conducted on a Nvidia V100 GPU with 16 GB of VRAM. For gradient descent, we employ Riemannian Adam [103] with an initial learning rate of 0.01 and standard β values of 0.9 and 0.999. The other hyper-parameters were selected based on the best performance on the validation set ($\mathcal{D}_{val}$) under the given computational constraints. In our experiments, we empirically set $k = 2, d = 32, h = 4$, and $\eta = 10$. Algorithm 6 further details our meta-learning procedure and the implementation code with our experiments is available at `https://anonymous.4open.science/r/HGRAM-5F32`.

5.4.4 Hyper-parameter tuning

We explore the following search space and tune our hyper-parameters for best performance. The number of tasks in each batch are varied among 4, 8, 16, 32 and 64. The learning rate

Algorithm 6: H-GRAM meta-learning algorithm

Input: Graphs $\mathcal{G}^{\cup} = \{\mathcal{G}_i\}_{i=1}^{\|\mathcal{G}^{\cup}\|}$, Ground truth Y;
Output: Predictor P_θ, parameters θ;

Initialize θ_* and θ as HGCN model and meta-update parameters, respectively ;
Partition graphs into node-centric subgraphs
$S_1, S_2, ...S_{\|V\|} = Partition\,(\mathcal{G}^{\cup})$;
Batch graphs into tasks for meta-learning
$\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_{\|\mathcal{T}\|}\}$;
while *not converged* **do**
  $\mathcal{T}_{train} \leftarrow sample(\mathcal{T})$;
  for $\mathcal{T}_i \in \mathcal{T}_{train}$ **do**
    # Batch of support and query set from the tasks
    $\{S_u\}^s, \{Y_u\}^s \leftarrow \mathcal{T}_i^s$;
    $\{S_u\}^q, \{Y_u\}^q \leftarrow \mathcal{T}_i^q$;
    for $j \in [1, \eta]$ **do**
      # Update θ^* using support set
      $h_u^s = HGCN_{\theta^*_{j-1}}({S_u}^s)$, via Eq. (5.1)
      $e_u^s = \frac{\sum_{i=1}^{\|V\|} \gamma_{iu} h_{iu}^s}{\sum_{i=1}^{\|V\|} \gamma_{iu}}$, via Eq. (5.2)
      $c_k^s = \frac{\sum_{Y_u = y_k} \gamma_i e_i^s}{\sum_{Y_u = y_k} \gamma_i}$, via Eq. (5.3)
      $p_k^s = \frac{e^{(-d_{\mathbb{H}}^c(e_u^s, c_k^s))}}{\sum_k e^{(-d_{\mathbb{H}}^c(e_u^s, c_k^s))}}$, via Eq. (5.4)
      $\mathcal{L}^s = \mathcal{L}(p^s, Y_u^s) = \sum_j y_i^s \log p_j^s$, via Eq. (5.6)
      $\theta_j^* \leftarrow exp_{\theta^*_{j-1}}^c(-\alpha \nabla \mathcal{L}^s)$
      # Record evaluation with θ^* on query set
      $h_u^q = HGCN_{\theta_j^*}({S_u}^q)$, via Eq. (5.1)
      $e_u^q = \frac{\sum_{i=1}^{\|V\|} \gamma_{iu} h_{iu}^q}{\sum_{i=1}^{\|V\|} \gamma_{iu}}$, via Eq. (5.2)
      $c_k^q = \frac{\sum_{Y_u = y_k} \gamma_i e_i^q}{\sum_{Y_u = y_k} \gamma_i}$, via Eq. (5.3)
      $p_k^q = \frac{e^{(-d_{\mathbb{H}}^c(e_u^q, c_k^q))}}{\sum_k e^{(-d_{\mathbb{H}}^c(e_u^q, c_k^q))}}$, via Eq. (5.4)
      $\mathcal{L}_{ij}^q = \mathcal{L}(p^q, Y_u^q) = \sum_j y_i^q \log p_j^q$, via Eq. (5.6)
    end
  end
  # Update meta-learning parameter θ
  $\theta \leftarrow exp_\theta^c(-\beta \nabla \sum_i \mathcal{L}_{iu}^q)$
end

Table 5.1: Hyper-parameter setup of real-world datasets. The columns present the number of tasks in each batch (# Tasks), HNN update learning rate (α), meta update learning rate (β) and size of hidden dimensions (d).

Dataset	# Tasks	α	β	d
arxiv-ogbn	32	10^{-2}	10^{-3}	256
Tissue-PPI	4	10^{-2}	5×10^{-3}	128
Fold-PPI	16	5×10^{-3}	10^{-3}	128
FirstMM-DB	8	10^{-2}	5×10^{-4}	128
Tree-of-Life	8	5×10^{-3}	5×10^{-4}	256

explored for both HNN updates and meta updates are $10^{-2}, 5 \times 10^{-3}, 10^{-3}$ and 5×10^{-4}. The size of hidden dimensions are selected from among 64, 128 and 256. The final best-performing hyper-parameter setup for real-world datasets is presented in Table 5.1.

5.5 Experimental Setup

Our experiments aim to evaluate the performance of the proposed H-GRAM model and investigate the following research questions:

RQ1: Does our hyperbolic meta-learning algorithm outperform the Euclidean baselines on various meta-learning problems?

RQ2: How does our model perform and scale in comparison to other HNN formulations in standard graph problems?

RQ3: How does H-GRAM model's performance vary with different few-shot settings, i.e., different values of k and N?

RQ4: What is the importance of different meta information components?

We use a set of standard benchmark datasets and baseline methods to compare the performance of H-GRAM on meta-learning and graph analysis tasks. The HNN models do not scale to the large datasets used in the meta-learning task, and hence, we limit our tests to Euclidean baselines. To compare against HNN models, we rely on standard node classification and link prediction on small datasets. Also, we do not consider other training learning paradigms, such as pretraining-finetuning and self-supervised learning because they require an exhaustive set of nodes and labels and do not handle disjoint problem settings.

5.5.1 Datasets

For the task of meta-learning, we utilize the experimental setup from earlier approaches [62]; two synthetic datasets to understand if H-GRAM is able to capture local graph information and five real-world datasets to evaluate our model's performance in a few-shot setting.

- **Synthetic Cycle** [62] contains multiple graphs with cycle as the basis with different topologies (House, Star, Diamond, and Fan) attached to the nodes on the cycle. The classes of the node are defined by their topology.
- **Synthetic BA** [62] uses Barabási-Albert (BA) graph as the basis with different topologies planted within it. The nodes are labeled using spectral clustering over the Graphlet Distribution Vector [102] of each node.
- **ogbn-arxiv** [60] is a large citation graph of papers, where titles are the node features and subject areas are the labels.
- **Tissue-PPI** [54, 161] contains multiple protein-protein interaction (PPI) networks collected from different tissues, gene signatures and ontology functions as features and labels, respectively.
- **FirstMM-DB** [92] is a standard 3D point cloud link prediction dataset.
- **Fold-PPI** [161] is a set of tissue PPI networks, where the node features and labels are the conjoint triad protein descriptor [108] and protein structures, respectively.
- **Tree-of-Life** [162] is a large collection of PPI networks, originating from different species.

For comparison with HNNs, we utilize the standard benchmark citation graphs of Cora [105], Pubmed [90], and Citeseer [105]. Table 5.2 provides more details of the datasets.

5.5.2 Baselines

We select the following baselines to understand H-GRAM's performance compared to state-of-the-art models in the tasks of meta-learning and standard graph processing.

- **Meta-Graph** [12], developed for few-shot link prediction over multiple graphs, utilizes VGAE [69] model with additional graph encoding signals.
- **Meta-GNN** [156] is a MAML algorithm developed over simple graph convolution (SGC) network [137].
- **FS-GIN** [140] runs Graph Isomorphism Network (GIN) on the entire graph and then uses the few-shot labelled nodes to propagate loss and learn.

Table 5.2: Dataset details. The columns present the dataset task (node classification or link prediction), number of graphs $|\mathcal{G}^{\cup}|$, nodes $|\boldsymbol{V}|$, edges $|\boldsymbol{E}|$, node features $|\boldsymbol{X}|$ and labels $|\boldsymbol{Y}|$. Node, Link and N/L indicates whether the datasets are used for node classification, link prediction or both, respectively.

Dataset	**Task**	$\|\mathcal{G}^{\cup}\|$	$\|\boldsymbol{V}\|$	$\|\boldsymbol{E}\|$	$\|\boldsymbol{X}\|$	$\|\boldsymbol{Y}\|$
Synth. Cycle	Node	10	11,476	19,687	-	17
Synth. BA	Node	10	2,000	7,647	-	10
ogbn-arxiv	Node	1	169,343	1,166,243	128	40
Tissue-PPI	Node	24	51,194	1,350,412	50	10
FirstMM-DB	Link	41	56,468	126,024	5	2
Fold-PPI	Node	144	274,606	3,666,563	512	29
Tree-of-Life	Link	1,840	1,450,633	8,762,166	-	2
Cora	N/L	1	2,708	5,429	1,433	7
Pubmed	N/L	1	19,717	44,338	500	3
Citeseer	N/L	1	3,312	4,732	3,703	6

- **FS-SGC** [137] is the same as FS-GIN but uses SGC instead of GIN as the GNN network.
- **ProtoNet** [112] learn a metric space over label prototypes to generalize over unseen classes.
- **MAML** [43] is a model-agnostic learning method that learns on multiple tasks to adapt the gradients faster on unseen tasks.
- **HMLP, HGCN and HAT** [17, 45, 50] are the hyperbolic variants of Euclidean multi-layer perceptron (MLP), Graph Convolution Network (GCN) and Attention (AT) networks that use hyperbolic gyrovector operations instead of the vector space model.

Note that not all the baselines are applicable on both node classification and link prediction. Hence, we compare our model against the baselines on applicable scenarios.

5.6 Experimental Results

We adopt the standard problem setting studied in the literature [62]. In the case of synthetic datasets, we use a 2-way setup for disjoint label problems, and for the shared label problems the cycle graph and Barabási–Albert (BA) graph have 17 and 10 labels, respectively. The evaluation of our model uses 5 and 10 gradient update steps in meta-training and meta-testing, respectively. In the case of real-world datasets, we use 3-shot and 16-shot setup for node classification and link prediction, respectively. For real-world disjoint labels problem, we use the 3-way classification setting. The evaluation of our model uses 20 and 10 gradient update steps in meta-training and meta-testing, respectively. In the case of Tissue-PPI

dataset, we perform each 2-way protein function task three times and average it over 10 iterations for the final result. In the case of link prediction task, we need to ensure the distinct nature of support and query set in all meta-training tasks. For this, we hold out a fixed set comprised of 30% and 70% of the edges as a preprocessing step for every graph for the support and query set, respectively.

Table 5.3: Performance of H-GRAM and the baselines on synthetic and real-world datasets. The top three rows define the task, problem setup (Single Graph (SG), Multiple Graphs (MG), Shared Labels (SL) or Disjoint Labels (DL)) and dataset. The problems with disjoint labels use a 2-way meta-learning setup, and in the case of shared labels, the cycle and BA graph have 17 and 10 labels, respectively. In our evaluation, we use 5 and 10 gradient update steps in meta-training and meta-testing, respectively. The columns present the average multi-class classification accuracy and 95% confidence interval over five-folds. Note that the baselines are only defined for certain tasks, "-" implies that the baseline is not defined for the task and setup. Meta-Graph is only defined for link prediction. The reason for the large confidence interval of synthetic datasets is provided in Section 5.6.2.

Task	**Node Classification**				**Node Classification**				**Node Classification**				**Node Classification**			**Link Prediction**	
Setup	$\langle SG, DL \rangle$				$\langle MG, SL \rangle$				$\langle MG, DL \rangle$				$\langle SG, DL \rangle$	$\langle MG, SL \rangle$	$\langle MG, DL \rangle$	$\langle MG, SL \rangle$	$\langle MG, SL \rangle$
Dataset	**Syn. Cycle**		**Syn. BA**		**Syn. Cycle**		**Syn. BA**		**Syn. Cycle**		**Syn. BA**		**ogbn-arxiv**	**Tissue-PPI**	**Fold-PPI**	**FirstMM-DB**	**Tree-of-Life**
Meta-Graph	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	.719±.018	.705±.004
Meta-GNN	.720±.191		.694±.098		-	-	-	-	-	-	-	-	.273±.107	-	-	-	-
FS-GIN	.684±.126		.749±.093		-	-	-	-	-	-	-	-	.336±.037	-	-	-	-
FS-SGC	.574±.081		.715±.088		-	-	-	-	-	-	-	-	.347±.004	-	-	-	-
ProtoNet	.821±.173		.858±.126		.282±.039		.657±.030		.749±.160		.866±.186		.372±.015	.546±.022	.382±.027	.779±.018	.697±.009
MAML	.842±.181		.848±.186		.511±.044		.726±.020		.653±.082		.844±.177		.389±.018	.745±.045	.482±.054	.758±.022	.719±.011
G-META	.872±.113		.867±.129		.542±.039		.734±.033		.767±.156		.867±.183		.451±.028	.768±.025	.561±.052	.784±.025	.722±.028
H-GRAM	**.883±.145**		**.873±.120**		**.555±.041**		**.746±.028**		**.779±.132**		**.888±.182**		**.472±.035**	**.786±.031**	**.584±.044**	**.804±.021**	**.742±.013**

5.6.1 RQ1: Performance of Meta-Learning

To analyze the meta-learning capability of H-GRAM, we compare it against previous approaches in this area on a standard evaluation setup. We consider two experimental setups inline with previous evaluation in the area [62]; (i) with synthetic datasets to analyze performance on different problem setups without altering the graph topology, and (ii) with real-world datasets to analyze performance for practical application. Based on the problem setup, the datasets are partitioned into node-centric subgraphs with corresonding root node's label as the subgraph's ground truth label. The subgraphs are subsequently batched into tasks which are further divided into support set and query set for meta-learning. The evaluation metric for both the tasks of node classification and link prediction is accuracy $\mathcal{A} = |Y = \hat{Y}|/|Y|$. For robust comparison, the metrics are computed over five-folds of validation splits in a 2-shot setting for node classification and 32-shot setting for link prediction. Table 5.3 presents the five-fold average and 95% confidence interval of our experiments on synthetic and real-world datasets, respectively.

From the results, we observe that H-GRAM consistently outperforms the baselines in the

area on a diverse set of datasets and meta-learning problem setups. For the disjoint labels setting, H-GRAM outperforms the best baseline in both the cases of single and multiple graphs. In the case of synthetic graphs, we observe that subgraph methods of H-GRAM and G-Meta outperform the entire graph encoding based approaches showing that subgraph methods are able to limit the over-smoothing problem [19] and improve performance. Also, we observe that meta-learning methods(ProtoNet and MAML) are unreliable in their results producing good results for some tasks and worse for others, whereas H-GRAM is consistently better across the board, hence, we conclude that using label prototypes to learn inductive biases and transferring them using MAML meta updates is a more robust technique. We note that H-GRAM, unlike previous HNN models is able to handle graphs with edges and nodes in the order of millions, as shown by the evident performance on large real-world datasets including ogbn-arxiv, Tissue-PPI, Fold-PPI, and Tree-of-Life. Our experiments clearly demonstrate the significant performance of H-GRAM in a wide-range of applications and prove the effectiveness of meta-learning in HNN models.

5.6.2 Large Confidence Interval in Synthetic dataset

In the results presented in Table 5.3, the reported mean demonstrates the predictive power of H-GRAM and the 95% confidence interval estimates the degree of uncertainty. The large confidence interval in the results on synthetic datasets is because in meta-testing, we only sampled two-labels in each fold. In some cases where the structure of the local subgraphs in meta-training is significantly different compared to meta-testing, our model has poor performance due to limited scope of knowledge transfer. We observe that, in the limited number of data split possibilities in synthetic datasets, there generally is a case where our model does not perform well which results in a larger confidence interval. Real-world datasets contain many more labels, and hence, we are able to sample more for meta-testing, e.g., 5 labels for 3-way classification. This reduces the possibility of atypical results, thus leading to smaller intervals.

Table 5.4: Comparison with HNN models on standard benchmarks. We compare the Single Graph, Shared Labels (SG,SL) setup of the H-GRAM model to the baselines. The columns report the average multi-class classification accuracy and 95% confidence interval over five-folds on the tasks of node classification (Node) and link prediction (Link) in the standard citation graphs.

Dataset	**Cora**		**Pubmed**		**Citeseer**	
Task	**Node**	**Link**	**Node**	**Link**	**Node**	**Link**
HMLP	.754±.029	.765±.047	.657±.045	.848±.038	.879±.078	.877±.090
HAT	.796±.036	**.792±.038**	.681±.034	.908±.038	.939±.034	.922±.036
HGCN	.779±.026	.789±.030	.696±.029	**.914±.031**	**.950±.032**	.928±.030
H-GRAM	**.827±.037**	.790±.026	**.716±.029**	.896±.025	.924±.033	**.936±.030**

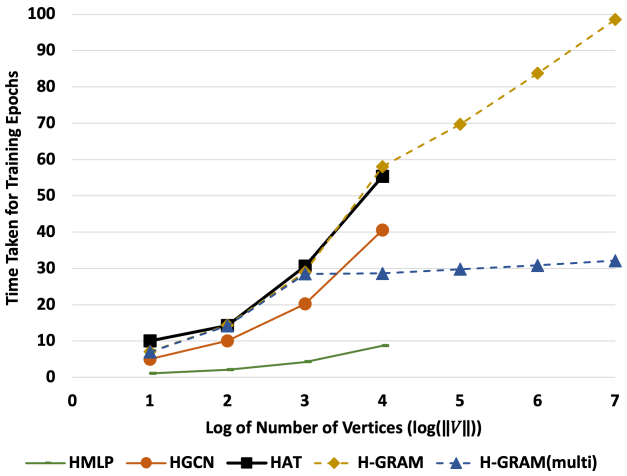
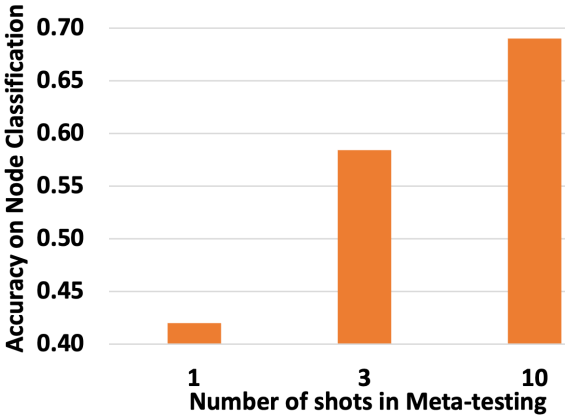


Figure 5.3: Time taken (per epoch) by H-GRAM in comparison to different HNN models for varying number of nodes $|\mathcal{V}| = \{\mathbf{10}^{\boldsymbol{i}}\}_{\boldsymbol{i=1}}^{\mathbf{7}}$ in Synth. BA graph. H-GRAM(multi) is the multi-GPU version of H-GRAM.

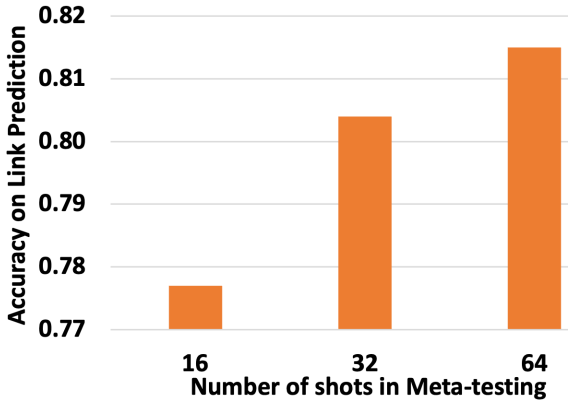
5.6.3 RQ2: Comparison with HNN models

The earlier HNN formulations of HMLP, HAT, and HGCN do not scale to large datasets, and hence, we were not able to compare them against H-GRAM on large-scale datasets. However, it is necessary to compare the standard HNN formulations with H-GRAM to understand the importance of subgraph encoders and meta-learning. Thus, we utilize the single graph and shared labels setup of H-GRAM to evaluate its performance on citation networks of Cora, Pubmed, and Citeseer for both tasks of node classification and link prediction. Additionally, we also compare the time taken by our model and other HNN models on varying number of nodes ($|\mathcal{V} = 10^{i}_{i=1}^{7}|$) in the Synthetic BA graph. For this experiment, we also consider a multi-GPU version of H-GRAM that parallelizes the HNN update computations and accumulates them for meta update.

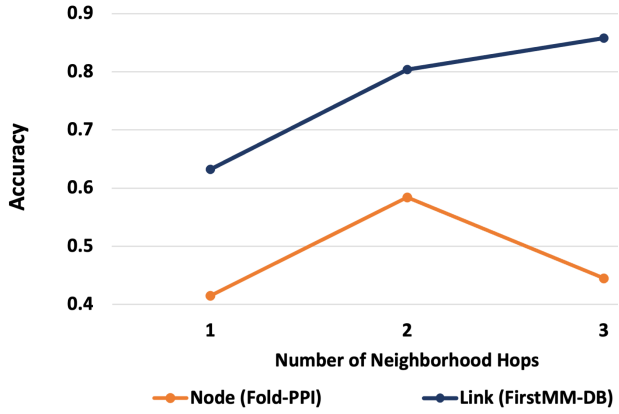
From our results, presented in Table 5.4, we observe that H-GRAM is, on an average, able to outperform the best baseline.This shows that our formulation of HNN models using meta-learning over node-centric subgraphs is more effective than the traditional models, while also being more scalable over large datasets. The scalability allows for multi-GPU training and also translates the performance gains of HNNs to larger datasets. In the results provided in Figure 5.3, we observe that the time taken by the models is inline with their parameter complexity (HMLP$\leq$HGCN$\leq$HAT$\leq$H-GRAM). However, the traditional HNN models are not able to scale beyond $|\mathcal{V}| = 10^4$, whereas, H-GRAM is able to accommodate large graphs. Another point to note is that H-GRAM(multi) is able to parallelize well over multiple GPUs with its time taken showing stability after 10^4 nodes (which is the size of nodes that a single GPU can accommodate).



(a) Number of Shots (vs) Accuracy for node classification on Fold-PPI dataset.



(b) Number of Shots (vs) Accuracy for link prediction on FirstMM-DB dataset.



(c) Number of hops (vs) Accuracy on the task of node classification and link prediction.

Figure 5.4: Performance of H-GRAM on challenging few-shot settings. The reported accuracies are multi-class classification accuracy averaged over five-fold runs of our model.

5.6.4 RQ3: Challenging Few-shot Settings

To understand the effect of different few-shot learning scenarios, we vary the number of few-shots M and hops K in the neighborhood. For the experiment on few-shot classification, we consider the problems of node classification on Fold-PPI dataset and link prediction on FirstMM-DB dataset. In node classification and link prediction, we respectively vary $M = 1, 2, 3$ and $M = 16, 32, 64$ and calculate the corresponding accuracy and 95% confidence interval of H-GRAM. The results for this experiment are presented in Figures 5.4(a) and 5.4(b) for node classification and link prediction, respectively. To determine the effect of hops in the neighborhood, we vary $K = 1, 2, 3$ for the same problem setting and compute the corresponding performance of our model. The results for varying neighborhood sizes are reported in Figure 5.4(c).

In the results on varying the number of few-shots, we observe a linear trend in both the tasks of node classification and link prediction, i.e., a linear increase in H-GRAM's accuracy with an increase in the number of shots in meta-testing. Thus, we conclude that H-GRAM, like other generic learning models, performs better with increasing number of training samples. In the experiment on increasing the neighborhood size, we observe that in the task of node classification $K = 3$ shows the best performance, but in link prediction $K = 2$ has the best performance, with a significant drop in $K = 3$. Thus, for stability we choose $K = 2$ in our experimental setup. The trend in link prediction also shows that larger neighborhoods can lead to increase in noise, which inturn can negatively affect performance.

Table 5.5: Ablation Study. H-ProtoNet and H-MAML can be considered H-GRAM's model variants without meta updates and label prototypes, respectively. H-GRAM(HMLP) and H-GRAM(HAT) represents the variant of H-GRAM with HMLP and HAT as base, respectively. Our final model, presented in the last row, uses HGCN as the base model. The columns report the average multi-class classification accuracy and 95% confidence interval over five-folds on different tasks on real-world datasets.

Task	**Node Classification**			**Link Prediction**	
Setup	$\langle SG, DL \rangle$	$\langle MG, SL \rangle$	$\langle MG, DL \rangle$	$\langle MG, SL \rangle$	$\langle MG, SL \rangle$
Dataset	**ogbn-arxiv**	**Tissue-PPI**	**Fold-PPI**	**FirstMM-DB**	**Tree-of-Life**
H-ProtoNet	.389±.019	.559±.027	.398±.023	.799±.015	.716±.004
H-MAML	.407±.023	.762±.056	.502±.046	.777±.018	.739±.005
H-GRAM(HMLP)	.370±.036	.537±.044	.372±.036	.772±.028	.688±.019
H-GRAM(HAT)	.462±.032	.777±.028	.573±.048	.794±.023	.732±.021
H-GRAM (ours)	**.472±.035**	**.786±.031**	**.584±.044**	**.804±.021**	**.742±.013**

5.6.5 RQ4: Ablation Study

In this section, we aim to understand the contribution of the different components in our model. To this end, we compare variants of our model by (i) varying the base HNN model (HMLP, HAT, and HGCN), and (ii) deleting individual meta-learning components (H-ProtoNet implies H-GRAM without meta updates) and (H-MAML implies H-GRAM without prototypes). The model variants are compared on the real-world datasets and the results are presented in Table 5.5. The ablation study indicates that meta gradient updates and label prototypes contribute to ≈16% and ≈6% improvement in H-GRAM's performance, respectively. This clearly demonstrates the ability of label prototypes in encoding inductive biases and that of meta gradients in transferring the knowledge from meta-training to the meta-testing phase. Additionally, from our study on different HNN bases for H-GRAM, we note that the HGCN base outperforms the other bases of HMLP and HAT by ≈19% and ≈2%, respectively. Thus, we choose HGCN as the base in our final model.

5.7 Broader Impact

Our model has the potential to impact various applications that involve graph-structured data, such as social network analysis, bioinformatics, and recommendation systems. Furthermore, the ability to generalize information from subgraph partitions of large datasets can be especially beneficial for applications with limited labeled data, such as in the fields of healthcare and finance. Moreover, H-GRAM also addresses several challenges in HNNs, including inductive learning, elimination of over-smoothing, and few-shot learning. These capabilities can be used to improve the performance of HNNs in various tasks such as node classification, link prediction, and graph classification. However, it is important to note that this model also has certain limitations. In particular, H-GRAM is based on a specific type of hyperbolic space, which may not be applicable to certain types of graph-structured data, and there are some assumptions made in the proof of our theoretical results which may not hold in general. Additionally, the meta-learning setup may not be suitable for all types of tasks, and further research is needed to test the performance of H-GRAM on other types of tasks. As a future direction, it would be interesting to investigate the effect of inadequate local subgraphs, scalability of H-GRAM on even larger datasets and explore the effectiveness of H-GRAM on other types of tasks with temporal or multi-modal graph data.

5.8 Summary

In this chapter, we introduce H-GRAM, a scalable hyperbolic meta-learning model that is able to learn inductive biases from a support set and adapt to a query set with disjoint nodes, edges, and labels by transferring the knowledge. We have theoretically proven the

effectiveness of node-centric subgraph information in HNN models, and used that to formulate a meta-learning model that can scale over large datasets. Our model is able to handle challenging few-shot learning scenarios and also outperform the previous Euclidean baselines in the area of meta-learning. Additionally, unlike previous HNN models, H-GRAM is also able to scale to large graph datasets.

Chapter 6

Conclusion and Future Work

This Thesis work aims to extend the accessibility of knowledge graphs to non-expert users/institutions who will be able to utilize non-technical textual queries to search over the vast amount of information stored in knowledge graphs. The overall goal is achieved by solving four subproblems; (A) Effective logical reasoning over knowledge graphs, (B) Retrieval through reasoning over knowledge graphs using natural language queries, (C) Integrating the framework with multi-modal semantic and structural information from entities and knowledge graphs, respectively, and (D) Scaling the hyperbolic architectures using meta-learning over local subgraphs.

6.1 Conclusion

For improving the performance of logical reasoning over knowledge graphs, the model utilizes the hyperbolic space to capture hierarchical dependencies. The main contributions of the research are; (i) Formulate the KG representation learning problem as a self-supervised query reasoning problem to leverage positive first-order existential queries, (ii) Introduce Hyperboloid Embeddings (HypE), a self-supervised dynamic representation learning framework that learns hyperboloid representations of KG units in a Poincaré hyperball. This is motivated by the need for non-Euclidean geometries, (iii) Perform an extensive set of empirical studies across diverse set of real-world datasets to evaluate the performance of HypE against several state-of-the-art baseline methods on the downstream task of Anomaly Detection., and (iv) Visualize the HypE embeddings to clearly interpret and comprehend the representation space.

Towards utilizing the natural language queries for retrieval from knowledge graphs represented in the hyperbolic space, we develop the ANTHEM model. The main contributions of this research are; (i) 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; (ii) A mechanism that utilizes attention units' activation to understand the internal working of ANTHEM and explain its product search mechanism on sample queries; (iii) 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 and (iv) 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.

For integrating the semantic and structural information in a hybrid multimodal reasoning framework, we developed the TESH-GCN model. The contributions of the research are; (i) We introduce Text Enriched Sparse Hyperbolic Graph Convolutional Networks (TESH-GCN), which utilizes semantic signals from input nodes to extract the local neighborhood and global graph features from the adjacency tensor of the entire graph to aid the prediction task; (ii) To enable the coordination between semantic signals and sparse adjacency tensor, we reformulate the hyperbolic graph convolution to a linear operation that is able to leverage the sparsity of adjacency tensors to reduce the number of model parameters, training and inference times (in practice, for a graph with 10^5 nodes and 10^{-4} sparsity this reduces the memory consumption from 80GB to 1MB). To the best of our knowledge, no other method has utilized the nodes' semantic signals to extract both local and global graph features; (iii) Our unique integration mechanism, not only captures both graph and text information in TESH-GCN, but also, provides robustness against noise in the individual modalities; and (iv) We conduct extensive experiments on a diverse set of graphs to compare the performance of our model against the state-of-the-art approaches on link prediction and also provide an explainability method to better understand the internal workings of our model. Additionally, we shall also evaluate and improve the performance of our models on ranking and indexing through human evaluation.

Finally, to improve the scalability of hyperbolic models over large graphs, we develop a meta-learning framework called H-GRAM. The primary contributions of this approach are; (i) We theoretically prove that HNNs rely on the nodes' local neighborhood for evidence in prediction, as well as, formulate HNNs to encode node-centric local subgraphs with root nodes as the local origin using the locality of tangent space transformations; (ii) We develop Hyperbolic GRAph Meta Learner (H-GRAM), a novel method that learns meta information (as meta gradients and label protonets) from local subgraphs and generalize it to new graphs with a disjoint set of nodes, edges and labels. Our experiments show that H-GRAM can be used to generalize information from subgraph partitions of large datasets, thus, enabling scalability in hyperbolic models; and (iii) Our analysis on a diverse set of datasets demonstrates that our meta-learning setup also solves several challenges in HNNs including inductive learning, elimination of over-smoothing and few-shot learning in several challenging scenarios.

6.2 Future Work

The future of hyperbolic space in machine learning is promising, as it has shown to be a better representation for hierarchical data than Euclidean space. However, there are still several challenges that need to be addressed to fully realize the potential of hyperbolic networks.

One of the most significant challenges is the development of more complex objective functions for classification and regression tasks. Currently, hyperbolic distance is the only available objective function, which is equivalent to L1-norm. Further research is needed to explore more complex objective functions that can capture the nuances of real-world problems. Another challenge is the unstable training and non-closure of hyperbolic networks. This issue needs to be addressed to ensure that hyperbolic networks can be used in practical applications beyond representation learning. The development of stable hyperbolic gradient descent techniques is necessary to enable effective training of hyperbolic networks. Furthermore, the current lack of hyperbolic libraries and standardization in hyperbolic network theory and architectures is a hindrance to the wider adoption of hyperbolic space in machine learning. The development of more scalable formulations for GPUs and the abstraction of hyperbolic network theory and architectures will help initiate new researchers in the area and facilitate the development of more complex models. In terms of future directions, hyperbolic space has the potential to impact various domains such as natural language processing, computer vision, and network analysis. Hierarchical information in natural language processing can be captured using dependency tree structures, and computer vision can benefit from the exponential volume growth of hyperbolic space. Additionally, hyperbolic space can be used to hierarchically aggregate information from network clusters to process high-level details.

In conclusion, hyperbolic space is a promising area of research in machine learning. While there are still challenges to be addressed, the future work in hyperbolic space involves the development of more complex objective functions, stable training techniques, scalable formulations, and standardization of hyperbolic network theory and architectures. Future research in hyperbolic space will likely have significant implications for various domains, and it is crucial to continue exploring the potential of hyperbolic networks.

6.3 Publications

1. **Nurendra Choudhary**, Edward W Huang, Karthik Subbian, Chandan K. Reddy, "Plug and Play Ensemble of Graph and Language Models for Improving Search Relevance in E-Commerce", ***Under Review***.

2. **Nurendra Choudhary**, Nikhil Rao, Chandan K. Reddy, "A Meta Learning Model for Scalable Hyperbolic Graph Neural Networks", ***Under Review***.

3. **Nurendra Choudhary**, Nikhil Rao, Karthik Subbian, Chandan K. Reddy, "Text Enriched Sparse Hyperbolic Graph Convolutional Networks", ***Under Review***.

4. Mehrdad Khatir, **Nurendra Choudhary**, Sutanay Choudhury, Khushbu Agarwal, Chandan K Reddy, "Pseudo-Poincaré: A Unification Framework for Euclidean and Hyperbolic Graph Neural Networks", International Joint Conferences on Artificial Intelligence 2023 (**IJCAI 2023**), ***Accepted***.

5. **Choudhary, Nurendra**, Nikhil Rao, Karthik Subbian, Srinivasan H. Sengamedu, and Chandan K. Reddy. "Hyperbolic Neural Networks: Theory, Architectures and Applications." In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 4778-4779. 2022., ***Tutorial @ KDD 2022 Published***.

6. **Choudhary, Nurendra**, Nikhil Rao, Karthik Subbian, and Chandan K. Reddy. "Graph-based Multilingual Language Model: Leveraging Product Relations for Search Relevance." In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 2789-2799. 2022. ***KDD 2022 Published***.

7. Vyas, Anoushka, **Nurendra Choudhary**, Mehrdad Khatir, and Chandan K. Reddy. "GraphZoo: A Development Toolkit for Graph Neural Networks with Hyperbolic Geometries." In Companion Proceedings of the Web Conference 2022, pp. 184-188. 2022. ***WWW 2022 Published***

8. **Choudhary, Nurendra**, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K. Reddy. "ANTHEM: Attentive Hyperbolic Entity Model for Product Search." In Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, pp. 161-171. 2022. ***WSDM 2022 Published***

9. **Choudhary, Nurendra**, Charu C. Aggarwal, Karthik Subbian, and Chandan K. Reddy. "Self-supervised Short-text Modeling through Auxiliary Context Generation." ACM Transactions on Intelligent Systems and Technology (TIST) 13, no. 3 (2022): 1-21. ***TIST 2021 Published***

10. **Choudhary, Nurendra**, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan Reddy. "Probabilistic entity representation model for reasoning over knowledge graphs." Advances in Neural Information Processing Systems 34 (2021): 23440-23451. ***NeurIPS 2021 Published***

11. **Choudhary, Nurendra**, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K. Reddy. "Self-supervised hyperboloid representations from logical queries over knowledge graphs." In Proceedings of the Web Conference 2021, pp. 1373-1384. 2021. ***WWW 2021 Published***

Bibliography

[1] A. Babenko and V. Lempitsky. The inverted multi-index. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(6):1247–1260, 2015.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2014. URL http://arxiv.org/abs/1409.0473.

[3] Ivana Balažević, Carl Allen, and Timothy Hospedales. Multi-relational poincaré graph embeddings. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2019. Curran Associates Inc.

[4] Gary Becigneul and Octavian-Eugen Ganea. Riemannian adaptive optimization methods. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=r1eiqi09K7.

[5] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[6] Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

[7] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia-a crystallization point for the web of data. *Journal of web semantics*, 7(3):154–165, 2009.

[8] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. doi: 10.1162/tacl_a_00051.

[9] Piero Andrea Bonatti, Stefan Decker, Axel Polleres, and Valentina Presutti. Knowledge Graphs: New Directions for Knowledge Representation on the Semantic Web (Dagstuhl Seminar 18371). *Dagstuhl Reports*, 8(9):29–111, 2019. ISSN 2192-5283. doi: 10.4230/DagRep.8.9.29. URL http://drops.dagstuhl.de/opus/volltexte/2019/10328.

[10] Silvère Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013. doi: 10.1109/TAC.2013.2254619.

[11] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems (NIPS)*, pages 1–9, 2013.

[12] Avishek Joey Bose, Ankit Jain, Piero Molino, and William L. Hamilton. Meta-graph: Few shot link prediction via meta learning, 2020. URL https://openreview.net/forum?id=BJepcaEtwB.

[13] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[14] James W Cannon, William J Floyd, Richard Kenyon, Walter R Parry, et al. Hyperbolic geometry. *Flavors of geometry*, 31:59–115, 1997.

[15] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 891–900, 2015.

[16] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. Toward an architecture for never-ending language learning. In *Aaai*, volume 5. Atlanta, 2010.

[17] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. In *Advances in neural information processing systems*, pages 4868–4879, 2019.

[18] Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. Low-dimensional hyperbolic knowledge graph embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6901–6914, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.617. URL https://aclanthology.org/2020.acl-main.617.

[19] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3438–3445, Apr. 2020. doi: 10.1609/aaai.v34i04.5747. URL https://ojs.aaai.org/index.php/AAAI/article/view/5747.

[20] Jie Chen and Yousef Saad. Dense subgraph extraction with application to community detection. *IEEE Transactions on Knowledge and Data Engineering*, 24(7):1216–1230, 2012. doi: 10.1109/TKDE.2010.271.

[21] Mingyang Chen, Wen Zhang, Wei Zhang, Qiang Chen, and Huajun Chen. Meta relational learning for few-shot link prediction in knowledge graphs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4208–4217, Hong Kong, China, November 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D19-1431.

[22] Francois Chollet et al. Keras, 2015. URL https://github.com/fchollet/keras.

[23] Nurendra Choudhary and Chandan K Reddy. Towards scalable hyperbolic neural networks using taylor series approximations. *arXiv preprint arXiv:2206.03610*, 2022.

[24] Nurendra Choudhary, Rajat Singh, Vijjini Anvesh Rao, and Manish Shrivastava. Twitter corpus of resource-scarce languages for sentiment analysis and multilingual emoji prediction. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1570–1577, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL https://aclanthology.org/C18-1133.

[25] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan Reddy. Probabilistic entity representation model for reasoning over knowledge graphs. *Advances in Neural Information Processing Systems*, 34:23440–23451, 2021.

[26] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K. Reddy. Self-supervised hyperboloid representations from logical queries over knowledge graphs. In *Proceedings of the Web Conference 2021*, WWW '21, page 1373–1384, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383127. doi: 10.1145/3442381.3449974. URL https://doi.org/10.1145/3442381.3449974.

[27] Nurendra Choudhary, Charu C Aggarwal, Karthik Subbian, and Chandan K Reddy. Self-supervised short-text modeling through auxiliary context generation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(3):1–21, 2022.

[28] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K. Reddy. ANTHEM: Attentive hyperbolic entity model for product search. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, WSDM '22, page 161–171, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450391320. doi: 10.1145/3488560.3498456. URL https://doi.org/10.1145/3488560.3498456.

[29] Nurendra Choudhary, Nikhil Rao, Karthik Subbian, and Chandan K Reddy. Graph-based multilingual language model: Leveraging product relations for search relevance. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2789–2799, 2022.

[30] Nurendra Choudhary, Nikhil Rao, Karthik Subbian, and Chandan K Reddy. Text enriched sparse hyperbolic graph convolutional networks. *arXiv preprint arXiv:2207.02368*, 2022.

[31] Joel E. Cohen. Infectious Diseases of Humans: Dynamics and Control. *JAMA*, 268 (23):3381–3381, 12 1992. ISSN 0098-7484. doi: 10.1001/jama.1992.03490230111047. URL https://doi.org/10.1001/jama.1992.03490230111047.

[32] Christopher De Sa, Albert Gu, Christopher Ré, and Frederic Sala. Representation tradeoffs for hyperbolic embeddings. *Proceedings of machine learning research*, 80: 4460, 2018.

[33] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 3844–3852, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

[34] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

[35] Bhuwan Dhingra, Christopher Shallue, Mohammad Norouzi, Andrew Dai, and George Dahl. Embedding text in hyperbolic spaces. In *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)*, pages 59–69, New Orleans, Louisiana, USA, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-1708. URL https://aclanthology.org/W18-1708.

[36] Fernando Diaz, Bhaskar Mitra, and Nick Craswell. Query expansion with locally-trained word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

[37] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2051–2060, 2017.

[38] Xin Luna Dong, Xiang He, Andrey Kan, Xian Li, Yan Liang, Jun Ma, Yifan Ethan Xu, Chenwei Zhang, Tong Zhao, Gabriel Blanco Saldana, et al. Autoknow: Self-driving knowledge collection for products of thousands of types. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2724–2734, 2020.

[39] Been Doshi-Velez, Finale; Kim. Towards a rigorous science of interpretable machine learning. In *eprint arXiv:1702.08608*, 2017.

[40] Yuchun Fang, Zhengyan Ma, Zhaoxiang Zhang, Xu-Yao Zhang, and Xiang Bai. Dynamic multi-task learning with convolutional neural network. In *IJCAI*, pages 1668–1674, 2017. URL https://doi.org/10.24963/ijcai.2017/231.

[41] Karoline Faust, Pierre Dupont, Jérôme Callut, and Jacques van Helden. Pathway discovery in metabolic networks by subgraph extraction. *Bioinformatics*, 26(9):1211–1218, 03 2010. ISSN 1367-4803. doi: 10.1093/bioinformatics/btq105. URL https://doi.org/10.1093/bioinformatics/btq105.

[42] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. CodeBERT: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1536–1547, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.139. URL https://aclanthology.org/2020.findings-emnlp.139.

[43] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1126–1135. JMLR.org, 2017.

[44] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. *MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding*, page 2331–2341. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450370233. URL https://doi.org/10.1145/3366423.3380297.

[45] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *Advances in neural information processing systems*, pages 5345–5355, 2018.

[46] Ning Gao, Hanna Ziesche, Ngo Anh Vien, Michael Volpp, and Gerhard Neumann. What matters for meta-learning vision regression tasks? In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14756–14766, 2022. doi: 10.1109/CVPR52688.2022.01436.

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

[48] M. Gromov. *Hyperbolic Groups*, pages 75–263. Springer New York, New York, NY, 1987. ISBN 978-1-4613-9586-7. doi: 10.1007/978-1-4613-9586-7_3. URL https://doi.org/10.1007/978-1-4613-9586-7_3.

[49] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[50] 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. Hyperbolic attention networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJxHsjRqFQ.

[51] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450340731.

[52] Jiafeng Guo, Yixing Fan, Xiang Ji, and Xueqi Cheng. 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*, SIGIR'19, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6172-9.

[53] Ruocheng Guo, Xiaoting Zhao, Adam Henderson, Liangjie Hong, and Huan Liu. Debiasing grid-based product search in e-commerce. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984.

[54] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

[55] Will Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding logical queries on knowledge graphs. In *Advances in neural information processing systems*, pages 2026–2037, 2018.

[56] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), December 2015. ISSN 2160-6455. doi: 10.1145/2827872. URL https://doi.org/10.1145/2827872.

[57] Ruining He and Julian J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In Jacqueline Bourdeau, Jim Hendler, Roger Nkambou, Ian Horrocks, and Ben Y. Zhao, editors, *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 507–517. ACM, 2016. doi: 10.1145/2872427.2883037. URL https://doi.org/10.1145/2872427.2883037.

[58] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[59] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, 2014.

[60] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

[61] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, WWW '20, page 2704–2710, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370233. doi: 10.1145/3366423.3380027. URL https://doi.org/10.1145/3366423.3380027.

[62] Kexin Huang and Marinka Zitnik. Graph meta learning via local subgraphs. *Advances in Neural Information Processing Systems*, 33:5862–5874, 2020.

[63] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *ACM International Conference on Information and Knowledge Management (CIKM)*, October 2013.

[64] Amir Jamaludin, Timor Kadir, and Andrew Zisserman. Self-supervised learning for spinal mris. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 294–302. Springer, 2017.

[65] Ming Ji, Yizhou Sun, Marina Danilevsky, Jiawei Han, and Jing Gao. Graph regularized transductive classification on heterogeneous information networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 570–586. Springer, 2010.

[66] B. Jia, C. Dong, Z. Chen, K. Chang, N. Sullivan, and G. Chen. Pattern discovery and anomaly detection via knowledge graph. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 2392–2399, 2018.

[67] Mehrdad Khatir, Nurendra Choudhary, Sutanay Choudhury, Khushbu Agarwal, and Chandan K Reddy. A unification framework for euclidean and hyperbolic graph neural networks. 2023.

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

[69] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.

[70] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

[71] Max Kochurov, Rasul Karimov, and Serge Kozlukov. Geoopt: Riemannian optimization in pytorch. *arXiv preprint arXiv:2005.02819*, 2020.

[72] Hung-yi Lee, Shang-Wen Li, and Thang Vu. Meta learning for natural language processing: A survey. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 666–684, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.49. URL https://aclanthology.org/2022.naacl-main.49.

[73] Jure Leskovec and Julian Mcauley. Learning to discover social circles in ego networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/7a614fd06c325499f1680b9896beedeb-Paper.pdf.

[74] Linfeng Li, Peng Wang, Jun Yan, Yao Wang, Simin Li, Jinpeng Jiang, Zhe Sun, Buzhou Tang, Tsung-Hui Chang, Shenghui Wang, and Yuting Liu. Real-world data medical knowledge graph: construction and applications. *Artificial Intelligence in Medicine*, 103:101817, 2020. ISSN 0933-3657. doi: https://doi.org/10.1016/j.artmed.2020.101817. URL https://www.sciencedirect.com/science/article/pii/S0933365719309546.

[75] Zheng Li, Mukul Kumar, William Headden, Bing Yin, Ying Wei, Yu Zhang, and Qiang Yang. 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)*, pages 2290–2301, 2020.

[76] Hailun Lin, Yong Liu, Weiping Wang, Yinliang Yue, and Zheng Lin. Learning entity and relation embeddings for knowledge resolution. *Procedia Computer Science*, 108: 345–354, 2017.

[77] Xi Victoria Lin, Richard Socher, and Caiming Xiong. Multi-hop knowledge graph reasoning with reward shaping. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3243–3253, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1362. URL https://www.aclweb.org/anthology/D18-1362.

[78] Lu Liu, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. *Learning to Propagate for Graph Meta-Learning*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[79] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. K-bert: Enabling language representation with knowledge graph. In *AAAI*, pages 2901–2908, 2020.

[80] Moshe Looks, Marcello Herreshoff, DeLesley Hutchins, and Peter Norvig. Deep learning with dynamic computation graphs. In *International Conference on Learning Representations*, 2017.

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

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

[83] Kurt Miller, Michael I Jordan, and Thomas L Griffiths. Nonparametric latent feature models for link prediction. In *Advances in neural information processing systems*, pages 1276–1284, 2009.

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

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

[86] Arsha Nagrani, Joon Son Chung, Samuel Albanie, and Andrew Zisserman. Disentangled speech embeddings using cross-modal self-supervision. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6829–6833. IEEE, 2020.

[87] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

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

[89] Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. Improving document ranking with dual word embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW '16 Companion, Republic and Canton of Geneva, CHE, 2016. International World Wide Web Conferences Steering Committee. ISBN 9781450341448.

[90] Galileo Mark Namata, Ben London, Lise Getoor, and Bert Huang. Query-driven active surveying for collective classification. In *International Workshop on Mining and Learning with Graphs*, Edinburgh, Scotland, 2012.

[91] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017.

[92] Marion Neumann, Plinio Moreno, Laura Antanas, Roman Garnett, and Kristian Kersting. Graph kernels for object category prediction in task-dependent robot grasping. In *Online proceedings of the eleventh workshop on mining and learning with graphs*, pages 0–6, 2013.

[93] Thanh Nguyen, Nikhil Rao, and Karthik Subbian. Learning robust models for e-commerce product search. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6861–6869, 2020.

[94] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, volume 11, pages 809–816, 2011.

[95] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 1955–1961. AAAI Press, 2016.

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

[97] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1ldO2EFPr.

[98] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 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.

[99] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. Text matching as image recognition. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16. AAAI Press, 2016.

[100] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[101] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2956-9. doi: 10.1145/2623330.2623732. URL http://doi.acm.org/10.1145/2623330.2623732.

[102] Nataša Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 2007.

[103] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ryQu7f-RZ.

[104] Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJgr4kSFDS.

[105] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 4292–4293. AAAI Press, 2015. ISBN 0262511290.

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

[107] Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3): 93–106, 2008.

[108] Juwen Shen, Jian Zhang, Xiaomin Luo, Weiliang Zhu, Kunqian Yu, Kaixian Chen, Yixue Li, and Hualiang Jiang. Predicting protein–protein interactions based only on sequences information. *Proceedings of the National Academy of Sciences*, 104(11): 4337–4341, 2007.

[109] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, page 373–374, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450327459. doi: 10.1145/2567948.2577348. URL https://doi.org/10.1145/2567948.2577348.

[110] Longxiang Shi, Shijian Li, Xiaoran Yang, Jiaheng Qi, Gang Pan, and Binbin Zhou. Semantic health knowledge graph: semantic integration of heterogeneous medical knowledge and services. *BioMed research international*, 2017.

[111] Ryohei Shimizu, Yusuke Mukuta, and Tatsuya Harada. Hyperbolic neural networks++. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ec85b0tUwbA.

[112] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/cb8da6767461f2812ae4290eac7cbc42-Paper.pdf.

[113] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *arXiv preprint arXiv:2004.09297*, 2020.

[114] Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. A graph-to-sequence model for AMR-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1150. URL https://www.aclweb.org/anthology/P18-1150.

[115] Ning Su, Jiyin He, Yiqun Liu, Min Zhang, and Shaoping Ma. User intent, behaviour, and perceived satisfaction in product search. In *Proceedings of the 11th International Conference on Web Search and Data Mining*, WSDM '18, page 547–555, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355810. doi: 10.1145/3159652.3159714. URL https://doi.org/10.1145/3159652.3159714.

[116] Haitian Sun, Andrew O Arnold, Tania Bedrax-Weiss, Fernando Pereira, and William W Cohen. Faithful embeddings for knowledge base queries. *Advances in Neural Information Processing Systems*, 33, 2020.

[117] Jianing Sun, Zhaoyue Cheng, Saba Zuberi, Felipe Perez, and Maksims Volkovs. Hgcf: Hyperbolic graph convolution networks for collaborative filtering. In *Proceedings of the Web Conference 2021*, WWW '21, page 593–601, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383127. doi: 10.1145/3442381.3450101. URL https://doi.org/10.1145/3442381.3450101.

[118] Zequn Sun, Muhao Chen, Wei Hu, Chengming Wang, Jian Dai, and Wei Zhang. Knowledge association with hyperbolic knowledge graph embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, November 2020. Association for Computational Linguistics.

[119] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.

[120] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1174. URL https://www.aclweb.org/anthology/D15-1174.

[121] George Fletcher ulong Pei, Xin Du and Mykola Pechenizkiy. Dynamic network representation learning via gaussian embedding. *NeurIPS 2019 Workshop on Graph Representation Learning*, 2019.

[122] Abraham A Ungar. *Analytic hyperbolic geometry: Mathematical foundations and applications*. World Scientific, 2005.

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

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

[125] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18(2):77–95, 2002.

[126] Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. In *ICLR*, 2015. URL http://arxiv.org/abs/1412.6623.

[127] Luke Vilnis, Xiang Li, Shikhar Murty, and Andrew McCallum. Probabilistic embedding of knowledge graphs with box lattice measures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 263–272, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1025. URL https://www.aclweb.org/anthology/P18-1025.

[128] Anoushka Vyas, Nurendra Choudhary, Mehrdad Khatir, and Chandan K. Reddy. Graphzoo: A development toolkit for graph neural networks with hyperbolic geometries. In *Companion Proceedings of the Web Conference 2022*, WWW '22, page 184–188, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450391306. doi: 10.1145/3487553.3524241. URL https://doi.org/10.1145/3487553.3524241.

[129] Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. A deep architecture for semantic matching with multiple positional sentence representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16. AAAI Press, 2016.

[130] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016.

[131] Q. Wang, X. Liu, W. Liu, A. Liu, W. Liu, and T. Mei. Metasearch: Incremental product search via deep meta-learning. *IEEE Transactions on Image Processing*, 29: 7549–7564, 2020.

[132] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[133] Shen Wang, Xiaokai Wei, Cicero dos Santos, Zhiguo Wang, Ramesh Nallapati, Andrew Arnold, Bing Xiang, and S Yu Philip. H2kgat: Hierarchical hyperbolic knowledge graph attention network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[134] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 950–958, 2019.

[135] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The World Wide Web Conference*, WWW '19, page 2022–2032, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366748. doi: 10.1145/3308558.3313562. URL https://doi.org/10.1145/3308558.3313562.

[136] W. X. Wilcke, P. Bloem, V. de Boer, R. H. van t Veer, and F. A. H. van Harmelen. End-to-end entity classification on multimodal knowledge graphs, 2020.

[137] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6861–6871. PMLR, 2019.

[138] Liwei Wu, Hsiang-Fu Yu, Nikhil Rao, James Sharpnack, and Cho-Jui Hsieh. Graph dna: Deep neighborhood aware graph encoding for collaborative filtering. In *International Conference on Artificial Intelligence and Statistics*, pages 776–787. PMLR, 2020.

[139] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 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*, SIGIR '17, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450350228.

[140] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

[141] Xing Xu, Huimin Lu, Jingkuan Song, Yang Yang, Heng Tao Shen, and Xuelong Li. Ternary adversarial networks with self-supervision for zero-shot cross-modal retrieval. *IEEE transactions on cybernetics*, 50(6):2400–2413, 2019.

[142] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations*, 2015.

[143] Liu Yang, Qingyao Ai, J. Guo, and W. Croft. 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.

[144] Tianchi Yang, Linmei Hu, Chuan Shi, Houye Ji, Xiaoli Li, and Liqiang Nie. Hgat: Heterogeneous graph attention networks for semi-supervised short text classification. *ACM Transactions on Information Systems (TOIS)*, 39(3):1–29, 2021.

[145] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf.

[146] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01): 7370–7377, Jul. 2019. doi: 10.1609/aaai.v33i01.33017370. URL https://ojs.aaai.org/index.php/AAAI/article/view/4725.

[147] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.45. URL https://aclanthology.org/2021.naacl-main.45.

[148] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=OeWooOxFwDa.

[149] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/9d63484abb477c97640154d40595a3bb-Paper.pdf.

[150] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.

[151] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. GraphSAINT: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJe8pkHFwS.

[152] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '19, page 793–803, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330961. URL https://doi.org/10.1145/3292500.3330961.

[153] Chuxu Zhang, Huaxiu Yao, Chao Huang, Meng Jiang, Zhenhui Li, and Nitesh V Chawla. Few-shot knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3041–3048, 2020.

[154] Jiawei Zhang, Haopeng Zhang, Li Sun, and Congying Xia. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*, 2020.

[155] Ningyu Zhang, Zhen Bi, Xiaozhuan Liang, Siyuan Cheng, Haosen Hong, Shumin Deng, Qiang Zhang, Jiazhang Lian, and Huajun Chen. Ontoprotein: Protein pretraining with gene ontology embedding. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=yfe1VMYAXa4.

[156] Fan Zhou, Chengtai Cao, Kunpeng Zhang, Goce Trajcevski, Ting Zhong, and Ji Geng. Meta-gnn: On few-shot node classification in graph meta-learning. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 2357–2360, New York, NY, USA, 2019. Association for Computing

Machinery. ISBN 9781450369763. doi: 10.1145/3357384.3358106. URL https://doi.org/10.1145/3357384.3358106.

[157] Fan Zhou, Chengtai Cao, Goce Trajcevski, Kunpeng Zhang, Ting Zhong, and Ji Geng. Fast network alignment via graph meta-learning. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 686–695, 2020. doi: 10.1109/INFOCOM41043.2020.9155456.

[158] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020. ISSN 2666-6510. doi: https://doi.org/10.1016/j.aiopen.2021.01.001. URL https://www.sciencedirect.com/science/article/pii/S2666651021000012.

[159] Jason Zhu, Yanling Cui, Yuming Liu, Hao Sun, Xue Li, Markus Pelger, Tianqi Yang, Liangjie Zhang, Ruofei Zhang, and Huasha Zhao. Textgnn: Improving text encoder via graph neural network in sponsored search. In *Proceedings of the Web Conference 2021*, WWW '21, page 2848–2857, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383127. doi: 10.1145/3442381.3449842. URL https://doi.org/10.1145/3442381.3449842.

[160] Jun Zhu. Max-margin nonparametric latent feature models for link prediction. In *Proceedings of the 29th International Conference on Machine Learning*, ICML'12, page 1179–1186, Madison, WI, USA, 2012. Omnipress. ISBN 9781450312851.

[161] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017.

[162] Marinka Zitnik, Rok Sosič, Marcus W. Feldman, and Jure Leskovec. Evolution of resilience in protein interactomes across the tree of life. *Proceedings of the National Academy of Sciences*, 116(10):4426–4433, 2019. doi: 10.1073/pnas.1818013116. URL https://www.pnas.org/doi/abs/10.1073/pnas.1818013116.

[163] Justin Zobel and Alistair Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, 38(2):6–es, July 2006. ISSN 0360-0300.