Applied Intelligence (2026) 56:118
https://doi.org/10.1007/s10489-026-07108-1

EPHG-CR: embedding propagation for heterogeneous graphs with class refinement

Brucce Neves dos Santos[1] · **Ricardo Marcondes Marcacini**[1] · **Alípio Mário Jorge**[2,3] · **Ricardo Campos**[3,4,5] · **Solange Oliveira Rezende**[1]

Received: 18 January 2024 / Accepted: 8 January 2026

Abstract

Heterogeneous graphs can represent real-world problems in a way close to reality, supporting diverse types of vertices and edges. However, their inherent heterogeneity poses challenges in interpreting problem semantics. To address this, heterogeneous graph embedding, aiming to map graph elements to low-dimensional vectors, simplifies subsequent machine learning analysis. This approach has gained prominence in machine learning, fueling classification, recommendation, and similarity search applications. Embedding diverse data is essential for efficient data processing. Incorporating language models, like BERT, into heterogeneous graphs enhances semantic context capture, which is particularly useful when one vertex type represents text. Language models stand out in contextual representation, enriching graph vertex embeddings for various tasks. This paper proposes a novel approach to enhancing heterogeneous graph embeddings by combining language models and task class data. Our approach increases vector quality, accounting for graph structure, semantic textual information, and task labels. We compared our proposal with a language model in the aspect-based sentiment analysis task, demonstrating competitive results and, in some cases, a slight superiority. Furthermore, we explore applications of embeddings from auxiliary vertices in another task, highlighting another advantage of the approach over the language model.

Keywords Heterogeneous network · Transductive · Graph embedding · Opinion mining · Language model

1 Introduction

Heterogeneous graphs can represent problems more closely to their real semantic nature, as they can accommodate various types of vertices and edges. This flexibility makes them suitable for various applications, from social network analysis to academic network studies, e-commerce, and cybersecurity [1–8].

However, this diversity that characterizes heterogeneous graphs also brings challenges, especially when dealing with their intrinsic heterogeneity [1, 9]. Heterogeneity, manifested in different types of vertices and edges, can give rise

Ricardo Marcondes Marcacini, Alípio Mário Jorge, Ricardo Campos, Solange Oliveira Rezende contributed equally to this work.

✉ Brucce Neves dos Santos
brucce.neves@alumni.usp.br

Ricardo Marcondes Marcacini
ricardo.marcacini@icmc.usp.br

Alípio Mário Jorge
amjorge@fc.up.pt

Ricardo Campos
ricardo.campos@ubi.pt

Solange Oliveira Rezende
solange@icmc.usp.br

[1] Institute of Mathematics and Computer Sciences, University of São Paulo, Av. Trabalhador São Carlense, N. 400, São Carlos 13566-590, São Paulo, Brazil

[2] Faculty of Sciences, University of Porto, Rua do Campo Alegre, Porto 4169-007, Porto, Portugal

[3] LIAAD, INESC TEC, Rua Dr. Roberto Frias, Porto 4200-465, Portugal

[4] University of Beira Interior, R. Marquês de Ávila e Bolama, Covilhã 6201-001, Portugal

[5] Ci2, Smart Cities Research Center (IPTomar), Quinta do Contador - Estrada da Serra, Tomar 2300-313, Portugal

Published online: 28 February 2026

to varied semantic interpretations for the same problem, making analyzing and processing these graphs a complex task.

In this context, heterogeneous graph representation learning, known as heterogeneous graph embedding, plays a crucial role. This technique aims to map the vertices and edges of these graphs to low-dimensional vectors, simplifying subsequent processing and analysis through machine learning algorithms [9–11].

Heterogeneous graph embedding has gained attention in the machine learning and data science community, as these representations are important for various tasks, including classification, recommendation, similarity search, and many other highly relevant applications [9–17].

Representation learning and embeddings have advanced various fields of data science and artificial intelligence by creating data representations that capture essential and relevant information in a lower-dimensional space, facilitating complex information analysis and processing [18, 19].

Many current studies are directing their efforts toward capturing heterogeneous graphs' structure and semantic nuances, including neighborhoods, communities, paths, and subgraphs, and incorporating vertex attribute information. However, the primary limitation of these works arises when dealing with vertices lacking specific attributes.

While some methods excel in representing vertices with attributes, they face significant challenges when handling vertices lacking associated information due to data scarcity or their heterogeneous nature. This gap in attribute availability for all vertices can hinder the models' ability to incorporate comprehensive and accurate information.

One promising approach to heterogeneous graph embedding is incorporating pre-trained language models, such as BERT. These models effectively represent textual context and are considered state-of-the-art in Natural Language Processing (NLP) tasks [20].

By incorporating language models into heterogeneous graphs, we leverage their ability to capture semantic context and relationships in text. This is especially advantageous in scenarios where one vertex type represents textual information. With their contextual representation, language models can enhance the embeddings of graph vertices, making them more informative and effective in various tasks.

Thus, this work aims to enhance the representation of heterogeneous graphs by integrating language models, such as BERT, and specific task class information. The main motivation is to improve the quality of vector representations of the graph vertices, considering both the graph structure and the semantics of textual information associated with the vertices and task class labels.

The main goal is to develop an innovative method that combines label propagation techniques with language models, resulting in enhanced vector representations for the vertices of the heterogeneous graph. These representations have broad applicability and can be employed in various tasks, such as:

- Classification: Enhancing the ability to classify different types of vertices in a heterogeneous graph.
- Recommendation: Facilitating the generation of more precise recommendations based on relationships among the graph's vertices.
- Similarity Search: Allowing for a more effective search for similar vertices within the graph.
- Other Information Processing Tasks: Benefiting from enhanced representations to perform a variety of data processing tasks, leveraging the richness of information captured in the embeddings of the graph's vertices.

Essentially, the work aims to enhance the understanding and processing of heterogeneous graphs, enabling the application of these enhanced representations in a wide range of tasks that benefit from the analysis and comprehension of the complex relationships present in these graphs.

We have developed a method that combines label propagation techniques with language models, resulting in enhanced embeddings for the graph's vertices. These embeddings have wide-ranging applicability and can be used in various tasks. For example, they can be employed in the target task, focusing on the vertices of interest, or in other tasks that benefit from embeddings of auxiliary vertices.

2 Background and related work

Learning representations and embeddings have played an important role in advancing various fields of data science and artificial intelligence [9, 11, 21–23]. This technique involves creating data representations that capture essential and relevant information, often in a lower-dimensional space, which facilitates the analysis and processing of complex information [9, 11, 16, 21–23].

In this context, the creation of embeddings for heterogeneous graphs has garnered significant attention [9–17]. On the one hand, these heterogeneous graphs are recognized in the literature for their ability to adequately represent complex problems, approaching the semantic aspects closely aligned with the problem's reality. This explicit modeling of diverse relationships among various types of graph objects

has been well-documented in prior research [1–8]. On the other hand, the complex structure formed by different types of vertices and edges presents an even greater challenge than homogeneous graphs when it comes to graph embedding tasks [9, 11, 16].

In the following sections, we will address the studies related to graph embeddings (Section 2.1) and embedding techniques that use label propagation (Section 2.2).

2.1 Heterogeneous graph embedding

Wang et al. [9] categorized Heterogeneous Graph Embedding approaches into four groups: "Structure-preserved heterogeneous graph embedding", "Attribute-assisted heterogeneous graph embedding", "Application-oriented heterogeneous graph embedding" and "Dynamic heterogeneous graph embedding".

In "Structure-preserved heterogeneous graph embedding", the aim is to preserve the graph's structure and semantics during the embedding learning process. This approach involves three fundamental structures: edges, meta-paths, and subgraphs. Meta-paths, representing paths between two vertices, describe the semantic relationships among pairs of vertices, enabling embeddings to capture not only the topology but also the various semantics of the graph. Each of these structures has its algorithms. For example, PME [24] deals with different types of edges by creating different metric spaces for various relationships, allowing pairs of vertices to be similar based on these metric spaces.

Regarding meta-paths, an algorithm is metapath2vec [25], which employs meta-paths generated through random walks, resulting in sequences of vertices. For subgraphs, mg2vec [26] is one such algorithm, using meta-graphs and vertices in the embedding learning process. It enumerates meta-graphs and preserves proximity between meta-graphs and vertices. In addition to these cited works, there are many others in this group [9, 12–14].

Apart from graph structure, attributes are another essential component. Thus, "Attribute-assisted heterogeneous graph embedding" creates embeddings based on the graph structure and aims to incorporate various attributes in the heterogeneous graph into these embeddings. One challenge of this approach is the heterogeneity of attributes, which led to the development of Heterogeneous Graph Neural Networks (HGNN), which can be categorized as unsupervised and semi-supervised. Unsupervised HGNNs include HetGNN [26], which is comprised of three parts. The first part focuses on aggregating content from different vertices to form a combined embedding. The second part is the neighborhood aggregator, which aggregates vertices of the same type. The third part consists of the type aggregator, utilizing an attention mechanism to aggregate embeddings from different types to form the final embedding.

On the other hand, semi-supervised approaches aim to learn embeddings for a specific task in an end-to-end manner, more frequently using attention mechanisms to capture relevant information. Among the various works in this group [9, 15, 16], the Heterogeneous graph Attention Network (HAN) [27] captures both the vertex's importance and semantic importance. It consists of three parts: vertex-level attention, semantic-level attention, and a task-focused layer that refines vertex embeddings based on a small number of labeled vertices.

Concerning applications of heterogeneous graph embeddings, the group "Application-oriented heterogeneous graph embedding" focuses on methods for applications such as classification, recommendation systems, and proximity search [9, 10]. The key points in this group are how to construct the graph for the application and how to incorporate domain-specific knowledge into the embeddings. For instance, recommendation systems can be naturally modeled as a heterogeneous graph, with vertices representing users and items and other types of relationships and information. In classification tasks, three prominent categories are author identification, user identification, and collective classification. Author identification aims to identify the author of anonymous articles based on author and paper similarity. User identification focuses on identifying users of interest and disinterest in a given application. Finally, collective classification deals with modeling relationships between different types of vertices, which is the most challenging aspect of modeling vertex relationships.

The previous groups primarily deal with static graphs, but graphs can change over time in real-world scenarios. Algorithms in the "Dynamic heterogeneous graph embedding" group fall into incremental updates or full retraining. In the incremental update category, algorithms use existing embeddings to learn embeddings for new vertices at each timestamp. An example is DyHNE [28], which uses the perturbation matrix to learn embeddings based on the heterogeneous graph's characteristics, preserving both the semantic and structural aspects of the graph while updating vertex embeddings without retraining the entire model. In the full retraining category, algorithms retrain the model at each timestamp and use techniques to capture temporal information between timestamps. For example, DyHATR [17] employs hierarchical attention to capture temporal information by altering vertex embeddings at different timestamps.

Beyond these groups, other works do not fit within these categories, such as heterogeneous graphs focused on natural language processing (NLP) tasks. Many NLP tasks can be modeled as heterogeneous graphs. In cases where vertices or a type of vertex represent text, embeddings from language models better represent the text. Label propagation algorithms can propagate this information to the rest of the vertices.

2.2 Label Propagation

Belkin et al. [29] introduced a versatile regularization framework for transductive classification within graphs. While these approaches may have variations, they share two common properties in transductive classification methods [30]. Firstly, their estimated class confidence vectors should exhibit similarity when two vertices are connected in the graph. Secondly, the estimated class confidence vectors of labeled vertices should closely resemble the actual class information. This framework can be expressed in a generic form, as shown in Equation 1 [31].

$$Q(\mathrm{F}) = \frac{1}{2} \sum_{o_i, o_j \in \mathcal{O}} w_{o_i,o_j} \Omega(\mathrm{f}_{o_i}, \mathrm{f}_{o_j}) + \mu \sum_{o_i \in \mathcal{O}^L} \Omega'(\mathrm{f}_{o_i}, \mathrm{y}_{o_i}) \tag{1}$$

The first term $\Omega(.)$ in the framework's function is tasked with computing the similarity between information vectors for each pair of related objects within the graph. This similarity can be derived using distance or dissimilarity functions. Conversely, the second term $\Omega'(.)$, calculates the similarity between the information of labeled objects and their respective actual original information. In this context, "labeled" refers to objects that possess external information and propagate this information within the graph. Within this equation, w_{o_i,o_j} denotes the weight of the relationship between objects, while the parameter μ signifies the importance of the actual original information during the label propagation process.

Among the various proposed methods, the GNetMine algorithm, introduced by Ji et al. [30], has some interesting features and shares some similarities with the proposed method. GNetMine considers different levels of importance for vertices and the importance of labeled data. Equation 2 defines the GNetMine regularization function, where $\lambda_{(u,v)}$ denotes the importance level between vertices u and v, with $0 \leq \lambda_{(u,v)} \leq 1$. To prevent highly connected (popular) vertices from dominating the class vector confidence estimates, $d(.)$ aggregates the edge weights of all neighbors of a vertex u that belong to the same type of relationship[1] as (u, v). f_v represents the estimated class confidence vector of a vertex v, and y_u signifies the actual class information of a labeled vertex u. The confidence in the real class information of a labeled vertex u is determined by $\alpha_{(u)}$, where $0 < \alpha_{(u)} \leq 1$. The regularization function is a minimization problem aimed at deriving a class confidence matrix F, which reflects the estimated class confidence across the entire graph.

$$Q(\mathrm{F}) = \sum_{u,v \in V} \lambda_{(u,v)} w_{u,v} \left\| \frac{\mathrm{f}_u}{\sqrt{d(u,v)}} - \frac{\mathrm{f}_v}{\sqrt{d(v,u)}} \right\|^2 + \sum_{u \in V^L} \alpha_{(u)} (\mathrm{f}_u - \mathrm{y}_u) \tag{2}$$

The method of Duran et al. [32] is also closely related to our proposal. They divide their method into two steps: first, it learns a vector representation for each label (this label is an identifier, not related to the class) through messages propagating along the edges of the entire graph. The second step involves learning representations for the vertices based on these labels.

Zheng et al. [33] proposed another similar method that aims to iteratively reconstruct a vertex's embedding and then propagate it to its neighbors. This process is done so neighbors of the same type have their embeddings aggregated, and these embeddings are concatenated. Additionally, each type of neighbor has its importance, which will be used when reconstructing the embedding.

GRACE (GRAph Clustering with Embedding propagation) was proposed by Yang [34], aiming to generate embeddings focused on the clustering task using both vertex attributes and edges, i.e., the graph structure. Furthermore, this method is end-to-end, meaning it simultaneously learns the embedding and the cluster unsupervised. To achieve this, it obtains all vertex attributes, learns an embedding, and then propagates this embedding in the network to obtain structure-based embeddings. These embeddings are then used to reconstruct the original attributes of the vertices while optimizing a clustering structure using self-training.

Paulo et al. [35] investigate the propagation of embeddings using language models in the context of news events. This work shares a concept that is closely related to what we are proposing. However, there are notable differences, such as their label propagation equation, which does not consider class information during propagation. They consider equal importance assigned to all types of relationships. Additionally, they employ pre-training using the Masked Language Model task of BERT for the initial embeddings.

[1] In homogeneous graphs, all vertices share the same type of relationship, while heterogeneous graphs organize vertices into different relationships.

Using language models during the learning process in heterogeneous graphs, where one type of vertex represents textual information, is highly advantageous. Language models like BERT effectively capture textual context and are state-of-the-art in NLP tasks [20].

3 Embedding propagation for heterogeneous graphs with class refinement (EPHG-CR)

This section will present the details of our approach for embedding refinement. Our proposed method consists of two stages. The first stage is divided into two parallel steps. One aims to generate embeddings for all graph vertices, while the other focuses on generating labels for all graph vertices. Dividing these steps allows for the independent optimization of each.

After the first stage is complete, the second stage begins. Its objective is to refine the embeddings, considering the labels generated in the first stage. Figure 1 illustrates this process. The embeddings from the first stage, highlighted in shades of gray, indicate that the learning was performed without considering class or label information. On the other hand, the colors resulting from the second stage (end of the process) represent an embedding matrix that considers both the graph structure and label information, reflecting a more refined and comprehensive representation of the vertices. For example, vertices "a" and "b", which are originally associated with the red and blue labels, respectively, also receive a slight influence from the orange label due to their connections with neighboring vertices.

Before delving into our proposal, let us consider some essential aspects of graph creation. It is important to note that our objective is not to prescribe a specific graph topology, as our approach does not impose such limitations. Instead, we aim to outline critical characteristics that the graph must exhibit. This flexibility is crucial since graph creation is intricately linked to both the dataset being utilized and the specific objectives of the task.

For instance, when dealing with textual datasets, a graph can be constructed where vertices represent individual texts, and edges encode the similarity between these texts. Alternatively, one can opt for a bipartite graph structure, where one set of vertices corresponds to texts, and the other set corresponds to individual words. The connections between vertices, in this case, signify the presence of a word in a given text. Moreover, it is worth noting that additional layers can be introduced by extracting further information from the text, creating diverse vertex types and connections. These additional layers may encompass temporal information, named entities, keywords, and various other elements, depending on the specific needs of the task.

Regarding the edges within the graph, it is paramount to ensure that the weights or values associated with these edges are appropriately normalized, with values constrained within the range [0, 1]. This normalization is crucial because our approach draws inspiration from the Regularization Framework [29, 31].

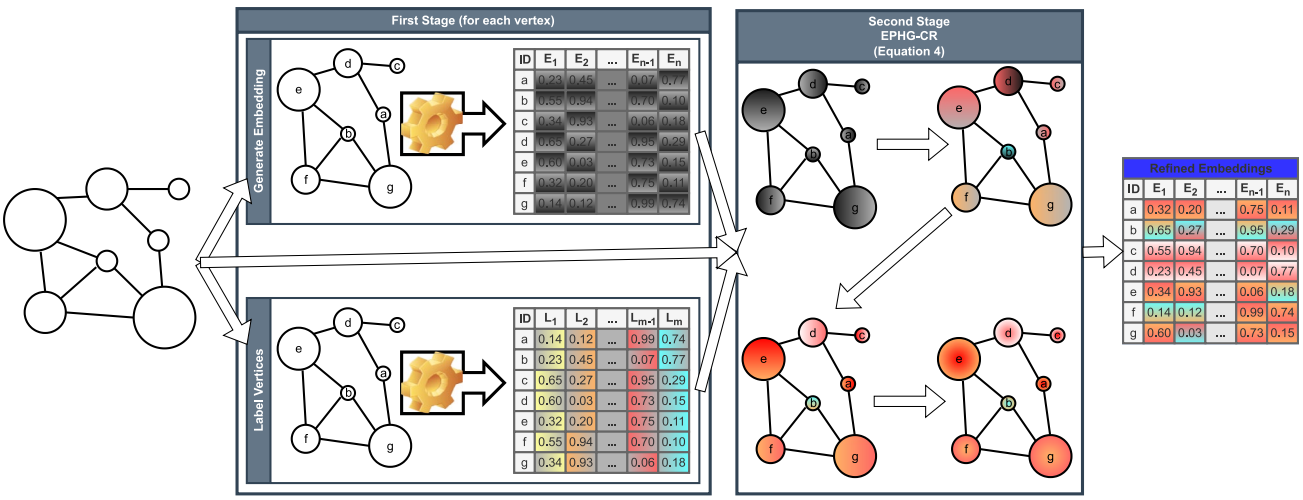

Fig. 1 Illustration of the proposal framework. The variable E_n represents the embedding dimension, while L_m denotes the number of associated labels

Section 3.1 will explain the process of generating embeddings for each vertex within the graph. Subsequently, Section 3.2 will explore various methods for generating labels for individual vertices within the graph. Finally, we will delve into the second stage of our proposal in Section 3.3, where we will elucidate the process of refining embeddings using these generated labels.

3.1 Node embedding generation

As mentioned earlier, the objective of this stage is to generate embeddings for each vertex of the graph. Various methods can generate embeddings from a graph, some of the most well-known ones being DeepWalk [36], Node2Vec [37], GraphSAGE [38], and Graph Convolutional Networks (GCN) [39]. These methods have advantages and disadvantages, along with others not mentioned here. However, regardless of these differences, they can be used as input for the proposal's second stage (Section 3.3). It is worth noting that we will not be evaluating the use of these models in this article, as it depends on the dataset and graph used in Section 4.

Models like BERT [40] can capture various types of text features, especially semantic and contextual information [41–44], due to their pre-training on a large corpus of text. In other words, Neural Language Models possess significant semantic and contextual information in their embeddings. However, despite having semantically and contextually rich embeddings, these embeddings are text-bound. They cannot represent objects related to the text, such as users, products, and other entities that can be modeled through graphs/networks. This is where the most significant advantage of graph/network representations lies – in making explicit the different types of relationships between various entity types [2, 3, 5, 30, 45–53].

Thus, this proposal stage aims to generate embeddings for all vertices using Language Model embeddings. To achieve this, it is necessary for the graph to have vertices representing text that can be utilized in the Language Model, such as reviews, keywords, or aspects. With textual vertices containing embeddings, the next step is to employ a label propagation algorithm to disseminate this embedding. In this regard, it is possible to utilize classical label propagation algorithms such as GFHF [54], LLGC [55], or GNetMine [30]. During this stage, when propagating language model embeddings throughout the graph, topology information is also integrated into the process. An important point is that the approach allows for mixing embeddings, meaning different vertices can receive embeddings from different sources. However, all embeddings must be the same size; our proposal cannot handle embeddings of different sizes.

3.2 Label generation

In this section, we will address the process of generating an initial set of labels for all vertices based on a set of target vertices, which are the focus of the task. Labeling large amounts of data is costly, so this approach relies on transductive learning. Transductive learning yields good results even with limited labeled data [51, 53, 56]. Therefore, it is possible to label a small subset of vertices manually. Additionally, depending on the application, using multi/cross-domain classifiers, unsupervised models, or even Large Language Models to label the target vertices is feasible.

Once a portion or all target vertices are labeled, label propagation is performed to assign labels to the remaining vertices. As mentioned earlier, this process can be carried out using classical algorithms. In the experimental evaluation, we will provide further details on how this step was executed (Section 4.4).

3.3 Regularization

In this section, we propose a novel approach, to the best of our knowledge, for refining embeddings based on vertex labels. As mentioned earlier, our approach takes as input a set of embeddings and a set of labels for each vertex to refine these embeddings through a label propagation process that considers the vertex labels.

Our approach is grounded in the Regularization Framework (Equation 1), where the primary objective is to minimize an objective function while satisfying two critical premises: (i) ensuring that the information of neighboring objects remains similar and (ii) ensuring that the information assigned to initialized objects during the classification process closely aligns with the actual information [57].

In a general context, consider $\mathcal{G} = \langle \mathcal{O}, \mathcal{R}, \mathcal{W}, \mathcal{E}, \mathcal{L} \rangle$ as a graph, where $\mathcal{O}$ represents the set of vertices formed by n subsets of vertices, i.e., $\mathcal{O} = \mathcal{O}^{C_1} \cup ... \cup \mathcal{O}^{C_n}$, where C is referred to as a layer. In this work, a layer is defined as a group of vertices of the same type or with a common purpose. Additionally, our proposal assumes the existence of a layer $\mathcal{O}^T$, which represents the subset of target vertices $T \in C$. In other words, it denotes the subset specifically targeted by the task. $\mathcal{R}$ represents the importance of different sets of edges, allowing us to increase or decrease the

weight of a group of edges during the propagation process. In other words, given two layers C_i and C_j, the importance of the edges connecting these layers is denoted as $\mathcal{R}^{C_i,C_j}$, where $\sum_i^C \sum_j^C \mathcal{R}^{i,j} = 1$. On the other hand, $\mathcal{W}$ indicates the weights of the edges between two objects. In contrast, $\mathcal{E}$ and $\mathcal{L}$ represent the embeddings and labels of each vertex, where $\mathcal{L}$ can be a one-hot matrix or contain the pertinence of a vertex for each of the classes.

Therefore, the objective of our proposal is to propagate the embedding $\mathcal{E}$ to the vertices $\mathcal{O}$ while weighting this information based on $\mathcal{W}$, $\mathcal{R}$, and $\mathcal{L}$. The process is illustrated by Equation 3, where λ_{o_i,o_j} represents the weighting or influence of the embedding $\mathcal{E}$ from vertex o_j on vertex o_i. In this context, $\mathcal{W}_{o_i,o_j}$ indicates the weight of the edge connecting vertices o_i and o_j. Additionally, $\mathcal{C}(o_i)$ denotes the layer to which vertex o_i belongs. Therefore, $\mathcal{O}^{\mathcal{C}(o_i)}$ returns the subset of vertices from the same layer as o_i. We use this to calculate the degree of vertex o_j, considering only the edges connecting vertices $o_k \in \mathcal{O}^{\mathcal{C}(o_i)}$. In other words, this part of the equation aims to penalize vertices with a high degree, considering only one layer of interest.

Furthermore, $\mathcal{R}^{\mathcal{C}(o_i),\mathcal{C}(o_j)}$ indicates the importance of the layers of vertices o_i and o_j, while $\mathcal{L}o_i, \text{argmax}(\mathcal{L}o_j)$ returns the pertinence of vertex o_i in the class of o_j, denoted by $\text{argmax}(\mathcal{L}o_j)$. This means that if vertex o_i has $\mathcal{L}o_i, \text{argmax}(\mathcal{L}_{o_j}) = 0$, then o_i will not receive any information from o_j.

$$\lambda_{o_i,o_j} = \frac{\mathcal{W}_{o_i,o_j}}{\sqrt{\sum_{o_k}^{\mathcal{O}^{\mathcal{C}(o_j)}} \mathcal{W}_{o_i,o_k}} \sqrt{\sum_{o_k}^{\mathcal{O}^{\mathcal{C}(o_i)}} \mathcal{W}_{o_j,o_k}}} \times \mathcal{R}^{\mathcal{C}(o_i),\mathcal{C}(o_j)} \times \mathcal{L}_{o_i,\text{argmax}(\mathcal{L}_{o_j})} \tag{3}$$

When we adapt the Regularization Framework (Equation 1) to our proposal, we obtain Equation 4. In this equation, $\sum_k \mathcal{L}_{o_i,k}$ calculates the sum of class pertinence for o_i and uses it as a normalizer. Meanwhile, the second part of the equation computes the similarity between the embeddings from the first stage e, and the embedding being refined f. The parameter μ indicates the importance of the original embedding. This means that a higher value of μ preserves more of the original embedding's information, ranging from 0 (preserves nothing) to 1 (preserves everything, i.e., the embedding remains identical to the original). Equation 4 can be solved using closed-form solvers or iterative methods like label propagation. In the next section, we will experimentally evaluate the proposed method. Figure 2 illustrates the convergence behavior of this function across 40 iterations (epochs).

$$Q(\mathrm{F}) = \sum_{o_i \in \mathcal{O}} \frac{\sum_{o_j}^{\mathcal{O}} \lambda_{o_i,o_j} \left\| \mathrm{f}_{o_i} - \mathrm{f}_{o_j} \right\|^2}{\sum_k \mathcal{L}_{o_i,k}} + \mu \sum_{o_i \in \mathcal{O}} (\mathrm{f}_{o_i} - \mathrm{e}_{o_i}) \tag{4}$$

3.4 Proposed algorithm

In the previous sections, we detailed the core components of our approach. Section 3.1 described the embedding generation and propagation process, which aims to initialize embeddings for the target vertices and propagate them across the graph. Section 3.2 focused on the label generation and propagation process, where initial labels are assigned to target vertices and propagated through the graph. Section 3.3 introduced the second stage of our approach, where embeddings are refined using Equation 4, which leverages both the graph structure and the propagated labels.

In this section, we unify these components into a single cohesive algorithm. The steps are outlined in Algorithm 1, providing a comprehensive view of how these processes interact to produce refined embeddings tailored for heterogeneous graph tasks.

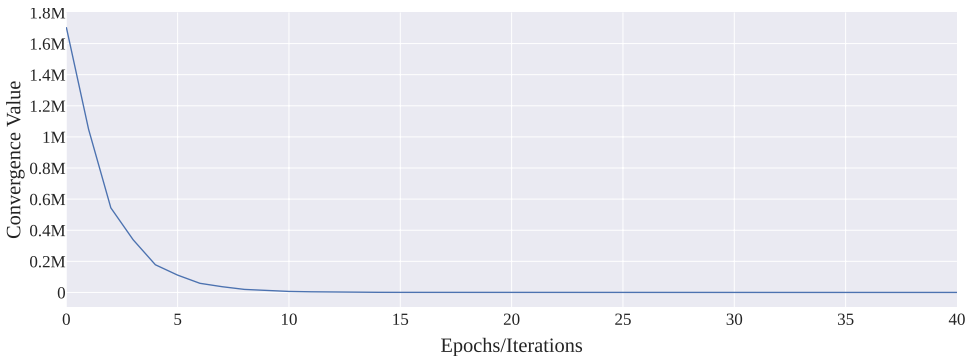

Fig. 2 Convergence Progress: Change in Convergence Metric Across Iterations

Algorithm 1 Embedding Propagation for Heterogeneous Graphs with Class Refinement (EPHG-CR).

Input:
- Graph $\mathcal{G} = \langle \mathcal{O}, \mathcal{R}, \mathcal{W} \rangle$
- Input data (for embeddings and labels)
- Label Propagation Algorithm (e.g., GNetMine, Label Propagation)
- Language Model (e.g., BERT)

Output:
- Refined embeddings

▷ **First Stage**

▷ **Step 1: Embedding Generation and Propagation**

for each target vertex v in $\mathcal{O}^E$ **do**
    Generate embedding $\mathcal{E}_v$ for vertex v using the language model
end for
Propagate the generated embeddings $\mathcal{E}$ throughout the graph $\mathcal{G}$ using the label propagation algorithm

▷ **Step 2: Label Generation and Propagation**

for each target vertex v in $\mathcal{O}^L$ **do**
    Assign label $\mathcal{L}_v$ to vertex v using a label source (e.g., ground truth, manual labels, or other sources)
end for
Propagate the labels $\mathcal{L}$ throughout the graph $\mathcal{G}$ using the label propagation algorithm

▷ **Second Stage**

Use Equations 3 and 4 to refine the embeddings generated in Step 1 with the labels propagated in Step 2

Return refined embeddings

In the proposed algorithm, the heterogeneous graph $\mathcal{G} = \langle \mathcal{O}, \mathcal{R}, \mathcal{W} \rangle$ consists of the following components: the vertices ($\mathcal{O}$), ($\mathcal{R}$) represents the importance of different sets of edges, and ($\mathcal{W}$) is the weights of the edges between two vertices.

The subset $\mathcal{O}^E$ represents the vertices used to generate embeddings through a language model (e.g., BERT), although other algorithms can also be applied for embedding generation. Similarly, the subset $\mathcal{O}^L$ consists of vertices that will be labeled using a label source, such as ground truth, manual annotations, or other methods.

4 Experimental evaluation

In this section, we will assess the proposed model's performance in the aspect-based sentiment analysis task and concurrently evaluate the quality of the embeddings generated by the model. The evaluation in the sentiment analysis task aims to gauge the effectiveness of the generated embeddings compared to the language model. On the other hand, the assessment of the embeddings will focus on determining their utility and effectiveness for other diverse tasks beyond aspect-based sentiment analysis.

4.1 Datasets

We will carry out these analyses using two aspect-based sentiment analysis datasets from SemEval 2014[2]. These datasets are described in Table 1 where "#Reviews" and "#Aspects" are the number of unique reviews and aspects, respectively.

Furthermore, the method proposed in Section 3 needs a graph (Section 4.2), an embedding (Section 4.3), and an algorithm for propagating labels and embeddings (Section 4.4). Finally, the proposed method will be evaluated in the sentiment analysis task in Section 4.5, while the embeddings will be evaluated in Section 4.6.

4.2 Creation of the graph

We describe how the vertices and edges were defined to build the graph and detail its characteristics, such as the number of vertices, the density of connections, and other relevant details.

Based on the sentiment analysis datasets described in Table 1, the graph we will be evaluating has five layers: five

[2] https://alt.qcri.org/semeval2014/task4/

Table 1 Overview of the selected datasets

			Train Sentiments			Test Sentiments			
Dataset	#Reviews	#Aspects	Neg	Neu	Pos	Neg	Neu	Pos	Total
Laptops	1900	1301	870	464	994	128	169	341	2966
Restaurants	2618	1637	807	637	2164	196	196	728	4728

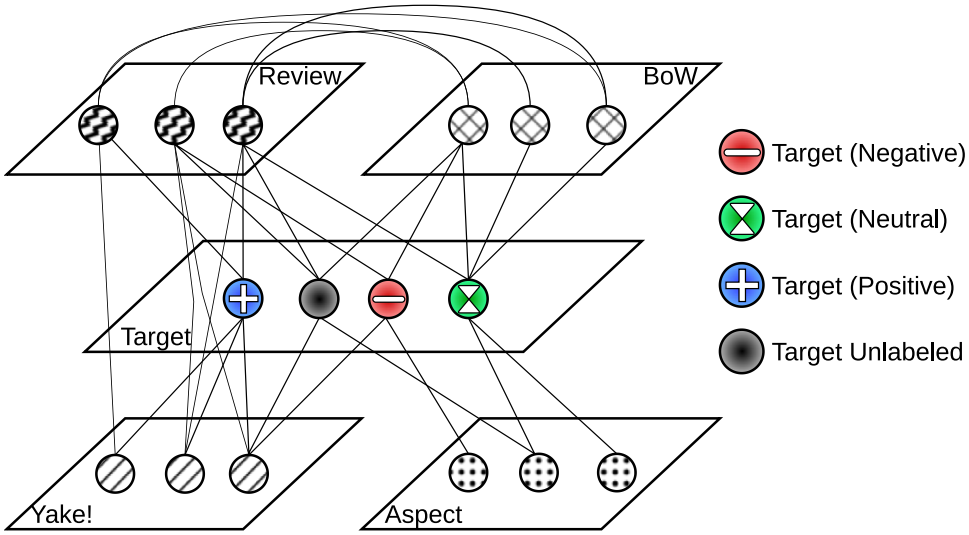

Fig. 3 Illustration of the proposed graph

sets of different vertices. Three of these layers are easily created, requiring little or no processing. However, the other two depend on external tools and may vary depending on their parameters. The complete graph that can be created using these five layers is illustrated in Fig. 3, and these layers are described below:

- **Target:** this is the main layer. It was created to compare the method proposed in the classification task with other algorithms. The size of this layer is the same as the dataset used. In other words, each vertex in this layer refers to an example of the dataset. This was necessary because the task is aspect-based sentiment analysis, which means a review can contain many aspects, and one aspect can be present in several reviews. So, creating a single vertex to represent the pair review and aspect was necessary. In addition, this layer is responsible for propagating the information in the graph since all other layers are connected directly or indirectly (through other layers that are connected to this one).
- **Aspect:** as previously mentioned, an aspect can be present in more than one dataset instance. The goal of this layer is to represent the dataset's unique aspects. For this layer, the unique aspects were considered without any preprocessing in the text; this means that aspects will be identified by how they appear in the text; for example "play", "playing" and "Play" are all different aspects. We chose not to perform any preprocessing to keep it simple and language-independent.
- **Review:** analogously to the Aspect layer, this layer aims to represent the unique reviews present in the different examples of the dataset. In addition, no preprocessing was performed on the reviews. An example would be the following opinion: "The price of the product is great, but the quality was disappointing". This opinion would be connected to two Target vertices: ("The price of the product is great, but the quality was disappointing", "Price") and ("The price of the product is great, but the quality was disappointing", "Quality").
- **BoW:** this layer, differing from the previous layers, was constructed based on the Bag-of-Words (BoW) representation of the dataset's reviews. Consequently, preprocessing steps for the text, such as stopword removal and conversion to lowercase, were necessitated. Additionally, *n*-grams were incorporated, with *n* ranging from 1 to 4. Any *n*-gram that appeared in only one review was subsequently excluded. To signify the presence or absence of an *n*-gram in a review, we utilized the frequency at which the *n*-gram occurred within the review. However, once the BoW representation was complete, we applied "sum normalization" to each row, ensuring values fell from 0 to 1.
- **YAKE!** [58]**:** similar to the Bag-of-Words (BoW) layer, this layer is also constructed based on the dataset's reviews, but the goal is to identify relevant keywords in user opinions. We used a keyword extraction tool called YAKE! to extract the top 20 keywords from each review using a maximum of 4-grams, setting the window size

to 1 and employing the "seqm" deduplication algorithm with a threshold of 0.9. These parameter settings were chosen because they are the defaults for this tool. Following the extraction of keywords from each review, we applied "MinMax Normalization" for scaling, following the guidelines provided by Campos et al. [58]. In the case of YAKE!, a lower score indicates greater relevance for a keyword within a specific review. To align with this, we inverted the scores to consider a keyword more relevant when closer to a value of 1. Finally, we established a structure analogous to the Bag-of-Words, which was utilized to remove keywords present in only one review and apply "sum normalization" to each line. To illustrate this layer, we can consider the following review: "The price of the product is great, but the quality was disappointing". We could extract keywords such as "price", "product", and "quality was disappointing" using the YAKE! extractor.

Except for the Target layer, the other layers can be constructed in different ways. This means that any changes in the construction of these layers will directly impact the final result of the proposed approach. However, this article will not delve into the analysis of this impact. This is because various factors, such as different parameters used in the employed techniques, various text preprocessing methods, and even the way edge weight normalization was performed, can generate different layers, leading to an exponential number of possible combinations and results. Addressing all possible variations in this study would be impractical. Table 2 shows the number of vertices in each layer with the preprocessing described above.

In practice, any subset of layers (Fig. 3) can be utilized. However, specific criteria have been established to define a valid subset for this article's evaluation scope. These criteria are outlined below:

1. Target Completeness: the graph should include all the vertices from the Target layer. No edges might connect specific vertices from the Target layer to the rest of the graph depending on the chosen subset. It would be meaningless to assess a vertex without altering its embedding during the propagation step.
2. Vertex Reachability: once the initial criterion is met, the goal is to ensure that all vertices from the "labeled" vertices are reachable. In this context, "labeled" refers to external information propagated in the graph, including labels or embeddings. While similar to the previous criterion, the focus here is not on missing vertices but on guaranteeing access to all present ones in the context of a proposed k-partite graph, which typically disallows connections between vertices within the same layer. However, there is an exception: if an unreached vertex is a Target vertex, an edge will be established between this Target vertex and the labeled Target vertex with the highest cosine similarity in their embeddings. This exception enables connectivity within the k-partite graph, even when it is typically restricted between vertices of the same layer.

Table 2 Overview of the number of vertices in each layer

Datasets	Aspects	BoW	Review	Target	Yake
Laptops	1301	16660	1900	2966	3995
Restaurants	1637	24493	2618	4728	6769

Using only the first criterion described earlier, we find 48 valid subgraphs, including some invalid subsets. For instance, one subset consists of two types of edges: the first type connects "Target" and "Aspect", while the second type connects "BoW" and "Review". This subset is invalid despite containing all "Target" vertices because no edges connect the "BoW" or "Review" layers to the "Target" layer. Using the second criterion, this is corrected, leaving 43 valid subgraphs.

Experimental evaluation of all 43 subgraphs is unfeasible for several reasons. First, the number of subgraphs imposes significant computational demands and time constraints. Second, exhaustive evaluation may not provide substantially different insights than assessing a representative subset. Moreover, resource limitations, including hardware and human resources, make it impractical to assess every subgraph comprehensively. Hence, we have evaluated five subgraphs as a more practical and efficient approach. Additionally, Table 3 will describe these selected subgraphs.

The five subgraphs for evaluation were carefully selected based on several factors. The first subgraph (A&R), composed only of the "Aspect-Target" and "Review-Target" relationships, was chosen as the most straightforward and fundamental starting point. This allows us to establish a reference baseline from which we can assess the impact of additional layers.

The second subgraph (A&R+BoW), which includes "BoW-Target" in the A&R subgraph, was selected to examine how the inclusion of the BoW layer influences graph connectivity and overall performance. The third subgraph (A&R+Yake), which adds "Yake-Target" to the previous combination, allows us to investigate the impact of a specific keyword extraction method on the graph.

The fourth subgraph (A&R+BoW+Yake), combining "BoW-Target" and "Yake-Target", is chosen to represent a more comprehensive configuration that reflects a realistic scenario of data and information propagation in the graph.

Table 3 Overview of the number of edges in each subgraph

		Restaurants		Laptops	
Subgraph	Relations	Vertices	Edges	Vertices	Edges
A&R	Aspect-Target Review-Target	8983	9456	6167	5932
A&R+BoW	Aspect-Target Review-Target BoW-Target	33476	119240	22827	76569
A&R+Yake	Aspect-Target Review-Target Yake-Target	15752	38590	10162	22716
A&R+BoW+Yake	Aspect-Target Review-Target BoW-Target Yake-Target	40245	148374	26822	93353
All Relations	All Relations	40245	203534	26822	132314

Finally, the "All Relations" graph, encompassing all proposed relationships, offers a holistic view of all possible interactions in the graph. Although computationally more challenging, its inclusion is valuable for understanding the cumulative influence of all layers and relationships.

The choice of these five subgraphs allows for a comprehensive analysis of different graph configurations, from the simplest to the most complex, helping to identify the most impactful relationships and the effects of different layers on information propagation.

4.2.1 Relationship of graph layers with sentiment analysis task

The construction of the heterogeneous graph aims to model the complexity of aspect-based sentiment analysis by using layers to capture semantic, textual, and structural information that provides a detailed representation of the data. Below, we detail how each layer contributes to this task.

- **Target:** represents the "review-aspect" pairs, connecting the other layers and enabling the propagation of semantic and topological information to identify the sentiment associated with each aspect.
- **Aspect:** facilitates the capture of shared semantic patterns across distinct contexts, allowing the model to better generalize the sentiment associated with each aspect. For example, an aspect like "price" may be evaluated negatively in multiple reviews, and this layer efficiently aggregates such information.
- **Review:** helps connect the targets by generating an overall opinion expressed in the review. It provides additional context, allowing the model to consider the general tone and other relevant textual information when determining the sentiment of a specific aspect.
- **BoW:** includes frequent *n*-grams from the reviews, capturing linguistic patterns that may indicate sentiment, and is used to connect the Targets.
- **YAKE!:** uses a keyword extraction algorithm to identify the most relevant terms, helping to identify semantic relationships that are important for connecting the Targets.

When integrated, these layers form a relational structure that captures the complex nuances of sentiments expressed in the analyses. The heterogeneous graph integrates data from different sources and types, such as text, aspects, and keywords, explicitly modeling their interactions. This approach allows for better information propagation, which enables the use of embeddings from these layers in other tasks, such as information retrieval.

4.3 Generation of the embeddings

We explain the process of generating the embeddings used for the graph vertices, discuss their dimensions, and discuss any preprocessing performed on them before they are used in the experiment.

We will analyze the proposed method using two types of embeddings: one from a language model and the other from a Random embedding. The chosen language model is BERT [40] pre-trained ("bert-base-cased"). The BERT embeddings are generated using a sentence pair structure. In other words, given a review $\mathrm{r} = (r_1, r_2, ..., r_S)$ and an aspect $\mathrm{a} = (a_1, a_2, ..., a_M)$, we employ a special [CLS] token at the beginning of the review and include a special [SEP] token after the review to connect it with the aspect tokens. Therefore, the input $\mathrm{x}^{r,a}$ is defined in the Equation 5:

$$\mathrm{x}^{r,a} = [CLS], r_1, r_2, ..., r_S, [SEP], a_1, a_2, ..., a_M, [SEP] \quad (5)$$

This input format will derive embeddings from BERT for the entire dataset, encompassing training and test examples. Since this model has not undergone fine-tuning, there is no issue in extracting embeddings for test examples.

Each token within the input will have its corresponding embedding among the embeddings provided by BERT as outputs for the input $\mathrm{x}^{r,a}$. For our experiments, we will utilize the embedding of the [CLS] token to represent the input $\mathrm{x}^{r,a}$.

Furthermore, to conduct specific experiments in Section 4.6, we will need to extract additional embeddings from BERT for each aspect. Since the embedding of the [CLS] token varies depending on the input used, this implies that different reviews with the same aspect a will have a different [CLS] embedding. Therefore, slightly modify the input to

obtain a unique embedding for each aspect. Equation 6 defines the input used to obtain the aspect embedding. The [CLS] token embedding will be utilized to represent the input.

$$\mathrm{x}^a = [CLS], a_1, a_2, ..., a_M, [SEP] \quad (6)$$

There are several options for generating random embeddings, with the most straightforward approach being creating one embedding for each example in the dataset. However, this approach poses two primary challenges. The first challenge is that the same review with different aspects could result in significantly different embeddings, and likewise, for different reviews with the same aspect. The second challenge is that this approach does not allow for creating an embedding for each aspect that captures similarity with examples in the dataset sharing that aspect, as explained earlier.

In this article, we have chosen to generate a Random embedding for each unique review and a separate embedding for each unique aspect in the dataset. Consequently, we summate the generated embeddings for each (review, aspect) pair in the dataset.

Regarding the size of the embeddings, we can generate random embeddings in any dimension. However, for consistency and comparability with BERT, we opted for a dimensionality of 768 ("random-768"), which matches the size used by BERT. Evaluating embeddings of varying dimensions could introduce unnecessary complexity and make isolating the impact of other experimental variables challenging.

An in-depth analysis of Random embeddings plays a critical role in our study as it allows us to assess the impact of the graph-based modeling techniques and approaches employed to enhance these embeddings. In contrast to BERT embeddings, which are enriched with linguistic knowledge through training on extensive text corpora, Random embeddings start with minimal knowledge. Therefore, we can gauge the added value of our enhancement strategies by examining how the graph and our approach can enrich these embeddings and make them more informative for specific tasks. This analysis provides crucial insights into the effectiveness of our approach and the substantial contribution that the graph structure can bring to compensate for the lack of intrinsic linguistic knowledge in Random embeddings.

4.4 Information propagation strategy

We present the adopted strategy for propagating labels and embeddings in the graph using the generated embeddings in Section 4.3. We describe the algorithm used to iteratively propagate the vertex information, including the specific parameters, such as the propagation rate and the stopping criterion.

As explained in Section 3, our approach consists of two stages of label propagation. In the first stage, we can employ any label propagation algorithm already in the literature. This initial stage aims to disseminate information, whether an embedding (E1) or a label (L1), to all vertices of the graph through the labeled vertices. In the second stage (EL2), we utilize the algorithm proposed in Equation 4, designed to propagate the embeddings while refining them based on the labels.

In the initial stage, where the propagation of embeddings (E1) and labels (L1) is conducted separately, we can employ distinct parameters and even utilize different label propagation algorithms as long as all vertices receive the propagated information. For our experiments, we have chosen to employ the GNetMine algorithm [30] for propagating both embeddings (E1) and labels (L1), albeit with varying hyperparameters. In the case of label propagation (L1), we will set $\mu^L = 1$, where the "L" designates its use in label propagation, thereby assigning equal significance to all relationship types. Conversely, for embedding propagation (E1), we will explore the hyperparameter μ^E across a range from 0.1 to 0.9, with the "E" indicating its role in embedding propagation while preserving uniform importance for all relationship types. The reason for not employing $\mu^E = 1$ is that it preserves the embeddings of labeled vertices unaltered, a result not aligned with the objectives of our experiments.

In the second stage (EL2), the proposed algorithm features parameters analogous to GNetMine's. Therefore, we will use μ^P ranging from 0.1 to 0.9, with "P" indicating the parameter used by the proposed method while maintaining equal importance for all relationship types. Furthermore, the proposed method takes the embeddings (E1) and labels (L1) propagated in the first stage as input. We will use the labels resulting from propagation (L1) with $\mu^L = 1$. As for the propagated embeddings (E1), we have used $\mu^E = \mu^P$ to simplify and reduce the number of experiments. This also allows us to analyze the impact of labels during the propagation of embeddings in the second stage, although, in practice, μ^E can differ from μ^P.

We will employ a common stopping criterion of 1000 iterations in both stages and a convergence threshold of 5×10^{-5}. These values have been selected based on prior research findings [53, 59]. While it is recognized that these hyperparameters can influence the outcome of information propagation, we have chosen not to analyze their impact.

To perform embedding propagation (E1) in the first stage, we will use the embeddings generated by BERT, as defined in Equation 5, for the vertices in the Target layer, both for training and testing. Since all vertices in the Target layer have embeddings, the second validity criterion for a subgraph, as defined in Section 4.2, was not applied since all

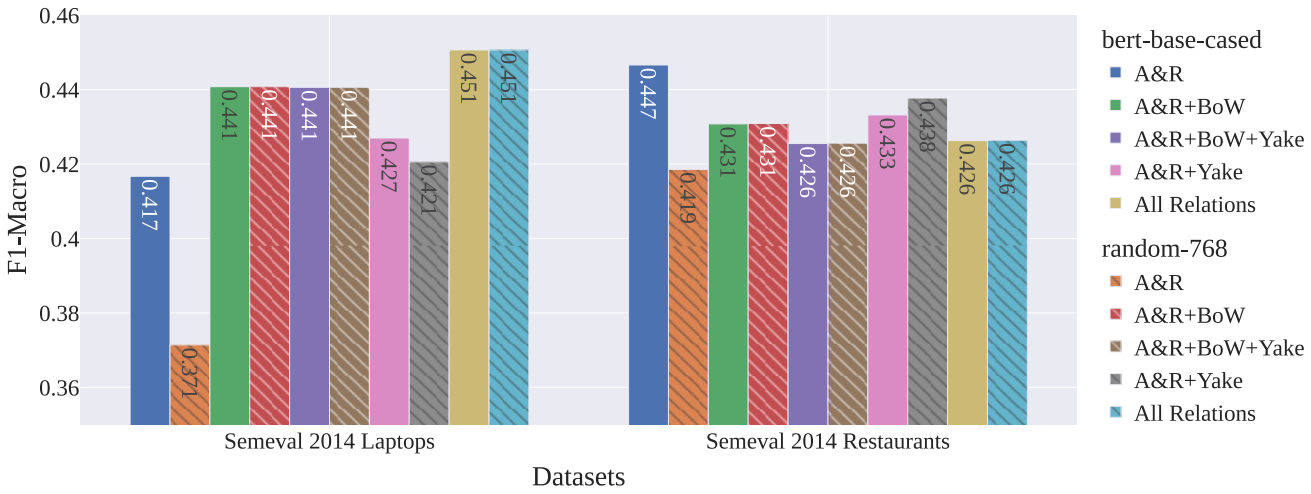

Fig. 4 Visualization of the impact of different edges connecting previously unreachable vertices. Higher values are better

vertices in the Target layer have embeddings (analogously for Random embeddings).

However, for label propagation (L1), we will only use the training vertices along with the labels provided by the dataset. In this case, applying the second graph validity criterion was necessary, as some test vertices may not be reachable. The similarity calculation between labeled vertices and those not reached depends on the embedding being propagated in the stage. This implies that the edges created based on the BERT embedding may differ from those created using the Random embedding. Furthermore, this choice will result in slight variations in the label propagation results depending on the embedding. We have chosen this approach to ensure fairness and to simulate the scenario properly where only a specific embedding is available.

To illustrate the performance difference, we will evaluate only the classification performance during label propagation (L1) using $\mu^L = 1$. Figure 4 illustrates the impact of establishing connections using BERT and Random embeddings generated across the various evaluated graphs. The classification performance using label propagation (L1) is generally low, as the assessed topologies struggle to effectively classify the vertices in the Target layer. However, it is essential to emphasize that the primary objective of this experiment is not focused on achieving optimal performance. Instead, the aim is to evaluate the outcomes when different types of embeddings are employed to establish connections among the unreached vertices within the Target layer. Notably, graphs containing the Bag-of-Words (BoW) layer exhibit identical results for both embeddings. This is attributed to the fact that these graphs do not require the creation of edges.

In contrast, connections established based on BERT embeddings tend to have a more significant impact on the overall graph performance for the remaining graphs compared to connections created using Random embeddings. However, in one particular instance, the performance of connections established by Random embeddings slightly surpasses those based on BERT embeddings.

4.5 Sentiment analysis

We have now completed the graph preparation, generated the embeddings, and selected the label propagation algorithms as discussed in the previous sections. We are ready to assess the performance of our approach in the aspect-based sentiment analysis task. We aim to measure how effectively the embedding refined by the proposed approach performs compared to the Strong Baseline model. It is essential to emphasize that our objective is not necessarily to surpass the Strong Baseline but to maintain a competitive performance. This is one way to evaluate the embedding quality produced by our proposal.

As the Strong Baseline method, we will perform fine-tuning of the BERT model[3], using the sentence pair approach, as described in Equation 5, with a learning rate of 5×10^{-5}. We chose BERT as the Strong Baseline method due to its widespread adoption and remarkable performance in natural language processing tasks. Furthermore, the initial embedding we are refining was derived from the pre-trained BERT itself, making it an appropriate benchmark to assess the impact of our refinement approach. This choice allows us to demonstrate the effectiveness of our technique and compare it with a well-established state-of-the-art method in aspect-based sentiment analysis.

We have chosen the F1-Macro to evaluate performance. This metric was selected due to the class imbalance in the

[3] "bert-base-cased"

datasets, where accuracy alone does not accurately reflect the results.

However, before we can compare the resulting embedding from our approach, we need to define which algorithm and hyperparameters we will use. Since the embedding is a numerical vector, a wide range of algorithms are available, ranging from classical ones like K-NN and SVM to more advanced techniques like GCN, GAT, and GNN. Although various algorithms could potentially be applied to evaluate the embeddings resulting from our approach, evaluating all these classifiers becomes computationally infeasible. Therefore, we chose to focus on a Multi-Layer Perceptron (MLP) for our experiments.

The selection of MLP is justified for several reasons. MLPs have shown versatility and adaptability in various machine learning tasks, making them a suitable choice for exploring the effectiveness of our refined embeddings. Additionally, MLPs can capture complex non-linear relationships in data, which may be advantageous in modeling the refined embeddings. Moreover, the choice of MLP aligns with the fact that BERT employs an MLP architecture, making it a relevant choice for evaluating our refined embeddings.

For this task, we will employ a MLP using the following hyperparameters: a learning rate of 0.001, the Adam optimizer, a maximum of 10,000 training epochs, and early stopping. Our primary focus will be evaluating the hidden layer's size and its impact on learning. We will keep all other hyperparameters at their default values as provided by the sci-kit library.

We assessed various configurations for the number of neurons in the hidden layer, specifically 100, 128, 256, 512, and 768. Each of these neural network architectures was trained with ten different random seeds to mitigate the effects of random weight initialization. For reference, we will denote a neural network as $NN^{(s,K)}$, where s represents the seed and K indicates the size of the hidden layer in the classifier.

In contrast to BERT, our experiments do not employ an MLP without hidden neurons. The critical distinction is that our approach aggregates information from the graph in the embedding. Consequently, we require a classifier with higher generalization capabilities. This broader generalization is essential because our embeddings undergo refinement with a wider focus. It enables our classifier to learn from the improved embeddings effectively, making them applicable to various tasks.

While we have chosen five classifiers for evaluation, it is challenging to examine their performance thoroughly, as we are dealing with numerous parameters for embedding refinement, resulting in various potential scenarios. Our analysis primarily focuses on identifying a classifier that demonstrates adequate generalization power and can reflect variations resulting from different configurations of embedding refinement parameters. We are not seeking the classifier with the best performance in a specific set of refinement parameters but rather the one that enables a more robust analysis of variations.

The following subsections will explore the results obtained through two approaches. Section 4.5.1 will be a statistical analysis using the Multiple Comparison Matrix tool. This evaluation will provide us with a broad and overall view of the results we achieved. Section 4.5.2 will delve deeper into our analysis with a detailed comparison of the results, examining each step that constitutes our approach.

4.5.1 Statistical analysis and classifiers performance

We will utilize the Multiple Comparison Matrix (MCM) introduced by Ismail-Fawaz et al. [60] to facilitate our analyses. This approach offers a unique perspective on statistical results without introducing new measures or evaluation metrics. Instead, it provides an alternative visualization method to address well-known challenges in ranking-based statistical analysis. The MCM enables us to compare each pair of evaluated objects, typically classifiers, across different datasets/tasks T and conduct statistical analyses on these pairs, ensuring a more equitable comparison and avoiding common issues associated with ranking-based approaches across all objects A. Furthermore, it maintains the consistency of statistical comparisons even when adding or removing objects A.

Understanding their layout and the information they contain is essential to facilitating the interpretation of the statistical heatmaps presented in the MCM. Each heatmap has dimensions of $m \times m$, where "m" represents the number of objects A being evaluated.

On the outer part of the heatmap are the names of each evaluated object A, with the average F1-Macro metric corresponding to that object A_i listed below it. Rows and columns in the heatmap are arranged based on these averages so that objects with higher F1-Macro averages are located to the left on the X-axis and closer to the top on the Y-axis, making it easier to identify the best-performing objects visually.

Comparison within the heatmap is conducted between the object A_i in the row and the object in the column A_j, denoted by MCM_{A_i,A_j}. Each cell of the heatmap is comprised of three values. The first value is the difference between the average F1-Macro metric of the row A_i and the average F1-Macro metric of the column A_j. This difference determines the color of the cell in the heatmap and provides a visual representation of performance differences.

The second value in the cell is in the format "win/tie/lose" and indicates how many times the row object achieved a higher score than the column object, how many times there was a tie, and how many times the row object had a lower score than the column object.

Lastly, the third value in the cell is the p-value derived from the Wilcoxon Signed Rank Test [61]. If the p-value is less than 0.05, this indicates a statistically significant difference between the evaluated objects. In such cases, the values in the cell will be displayed in bold and underlined to emphasize the statistical difference.

These details enable a comprehensive analysis of comparisons among different objects A, allowing us to identify which exhibit statistically significant performance differences. Our initial evaluation focuses on classifiers NN applied to tasks $T = E_{D,P,G,\mu}$. As defined previously, a classifier is represented by $NN^{(s,K)}$. However, for this analysis, we modify by incorporating the seed s as part of T, resulting in $T = E_{D,P,G,\mu} \times NN^{(s,K)}$, while $A = NN^{(K)}$. We have opted not to aggregate the results across different seeds within E because specific algorithms display significant standard deviations for a given E.

Figures 5 and 6 present the MCM for classifiers trained with BERT embeddings and Random embeddings, respectively. It is noticeable that in both cases, MLP(768) achieved the highest average F1-Macro score. In the scenario of Random embeddings, as the number of neurons increases, the average also increases. This trend is also consistent when considering BERT embeddings, except for MLP(256), which lags behind MLP(128); the others exhibit improvements. Regarding the comparison between BERT embeddings, there is no statistically significant difference among MLP(768), MLP(512), and MLP(128). However, considering that MLP(768) achieved the highest average score across both embeddings, it learned the task effectively through the different embeddings it was trained on.

The next MCM analysis will focus on μ, which, as mentioned in Section 4.4, indicates the extent to which the original information (in this case, the embedding) is retained in the labeled vertex set. In these figures, $A = \mu$, while $T = E_{D,P,G} \times NN^{(s,K)}$. This analysis examines how μ behaves across different datasets. Figure 7 illustrates the MCM for μ with BERT embeddings, while Figure 8 depicts the results for Random embeddings.

The optimal μ value is 0.3 for BERT embeddings, whereas for Random embeddings, it is 0.4. Furthermore, there is no statistically significant difference between the pairs (0.3, 0.2) and (0.4, 0.3) for BERT and Random embeddings. These results indicate that setting μ too high or too low can potentially hinder performance. In general, maintaining between 30% and 40% of the original information while letting the graph determine the remainder appears to be a suitable range. These are general statistics based on evaluating two datasets, both stages of the proposal, five classifiers, and various graphs. This analysis provides insights into how variations in this parameter can impact classifier performance.

Finally, let us analyze the overall performance of the graphs. Figures 9 and 10 display the MCM matrices. The first observation in Figure 10 is that depending on the parameters used to refine the embedding, the same classifier with the same seed can exhibit identical performance across different graph topologies. For instance, in the case of A&R+BoW and A&R+Yake, there are 111 ties in performance. This phenomenon also occurs in Figure 9, albeit on a smaller scale and among topologies that share at least one set of edges or layers.

Furthermore, in most cases, the average performance difference between the graphs was less than 1%, indicating that despite the statistical differences between some topologies, the overall results remained quite close. Lastly, it is worth highlighting that in the case of BERT embeddings, A&R+BoW, despite having a slightly higher average F1-Macro score than A&R+BoW+Yake, had more losses than wins. This suggests that the A&R+BoW+Yake topology is more stable despite the other having a slightly better score. A&R+BoW has some performance peaks that elevated its average score.

4.5.2 Detailed comparison of results and approach analysis

Moving on to a more detailed performance analysis, we will now select the best (✓) and worst (✗) results obtained from all possible combinations of parameters and hyperparameters. Unlike the previous statistical analyses, where we did not average over different seeds for the same classifier, we will perform averaging in this analysis. This results in 450 results for each pair of embeddings and datasets. For each dataset and embedding, we selected the classifier *NN*, μ, and graph G that address the questions based on the F1-Macro metric, as described below:

- What is the best and worst result obtained by stage E1? Illustrated in Table 4 as "✗ E1" and "✓ E1", respectively.
- What is the best and worst result obtained by stage EL2? Illustrated in Table 4 as "✗ EL2" and "✓ EL2", respectively.
- What result is obtained when we combine the best *NN* (Figs. 5 and 6), μ (Figs. 7 and 8), and G (Figs. 9 and 10)? Illustrated in Table 4 as "✓ Statistical".
- What result is obtained when we combine the worst *NN* (Figs. 5 and 6), μ (Figs. 7 and 8), and G (Figs. 9 and 10)? Illustrated in Table 4 as "✗ Statistical".

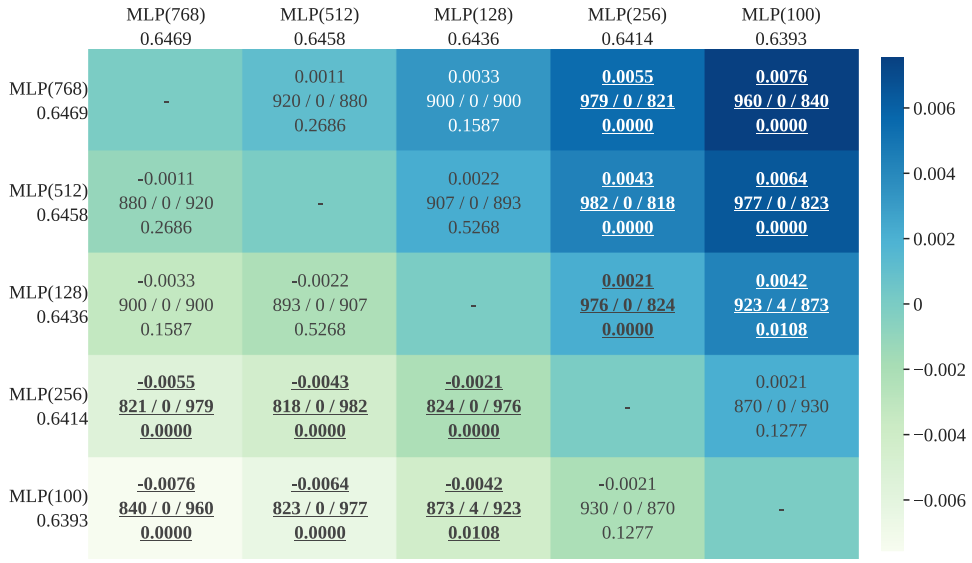

Fig. 5 MCM for Classifiers with BERT embeddings

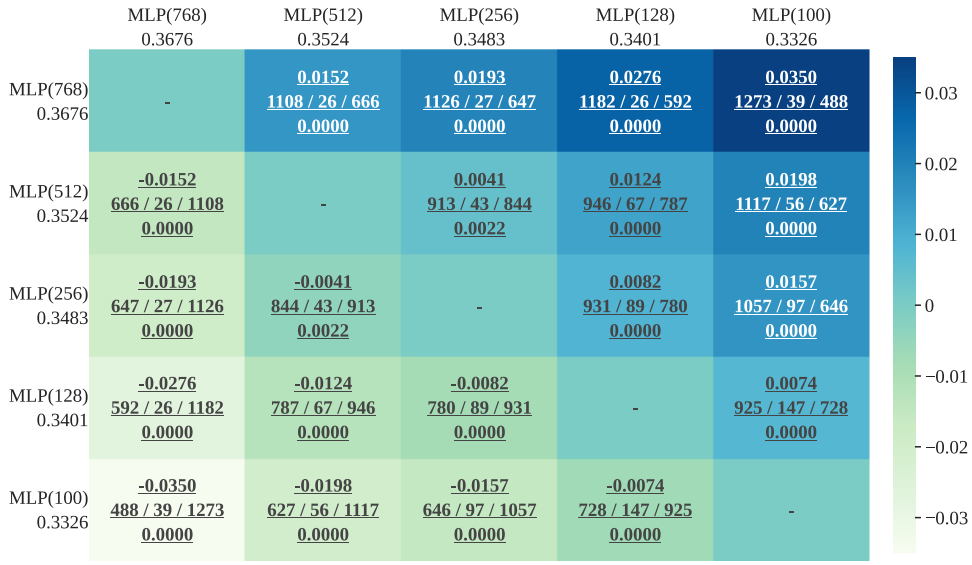

Fig. 6 MCM for Classifiers with Random embeddings

It is worth noting that the "✗ E1", "✓ E1", "✗ EL2" and "✓ EL2" values of *NN*, μ, and *G* change depending on the dataset and embedding. While "✓ Statistical" and "✗ Statistical" change depending on the embedding only.

Figures 11 and 12 provide the results obtained for the Laptops and Restaurants datasets. These results were generated using the models and parameters outlined in Table 4. Each figure contains bars representing different stages: the bars without marks represent the first stage (E1), while the bars with marks represent the second stage (EL2). Having both stages displayed for each configuration enables us to compare how the same parameters perform in both stages and draw meaningful conclusions.

As mentioned, our reference point is the fine-tuned BERT model ("bert-base-cased"). Within these figures, you will notice three lines representing distinct performance metrics. The top line indicates the best results achieved, the bottom line illustrates the worst results and the middle line showcases the average performance across ten runs. This approach helps us comprehend the overall performance range for each configuration.

For the Random embeddings ("random-768"), we trained MLP models using the unrefined Random embeddings. As some items in Table 4 involve various MLP topologies (items 7-12 and 19-24), we computed the baseline results based on five selected MLP topologies. These baselines were trained using the same ten random seeds as our other experiments.

Consequently, the top line represents the best performance achieved among the 50 classifiers trained for baseline purposes, while the bottom line indicates the worst

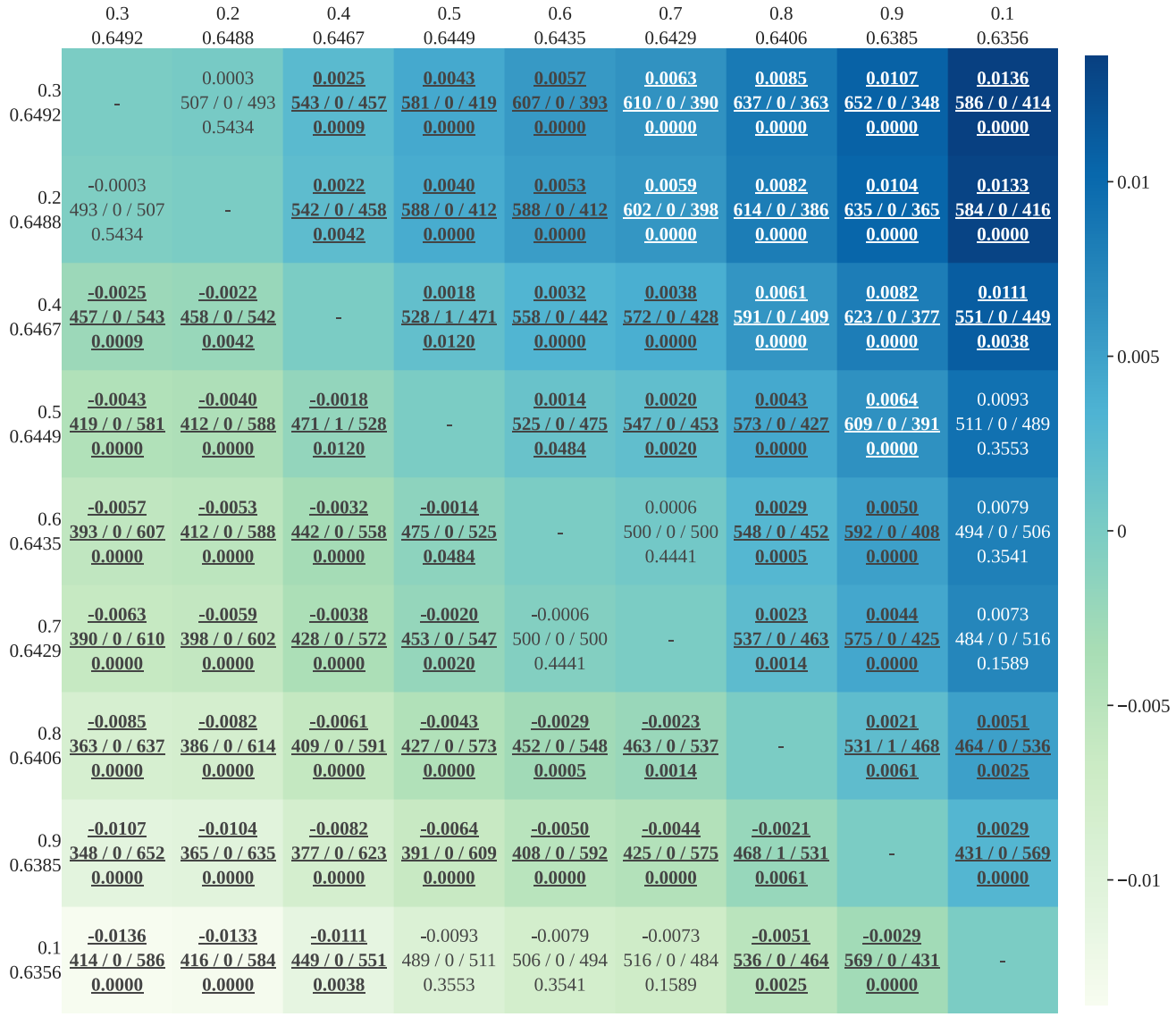

Fig. 7 MCM for μ with BERT embeddings

performance. The middle line represents the average performance across all 50 classifiers, providing a comprehensive view of the baseline results.

Figure 11 shows that the worst statistical result (✗ Statistical [ID: 1]) is slightly above the line, indicating the lowest value in the Strong Baseline. This suggests that even though this combination is not the best, it remains competitive. Except for ✗ E1 (ID: 3), all other worst scenarios (✗ odd IDs) illustrated in this figure are associated with MLPs with a small number of neurons. However, this does not necessarily mean that this configuration is entirely wrong, as it tends to perform well using the same setup in the other stage. This is evident when comparing ✗ E1 and ✓ E1 (ID: 4), which is the only case where the worst and best results differ due to using a different μ.

In Figure 12, we can observe that, unlike the Laptops dataset, the model failed to learn and predict all instances belonging to the majority class in the worst scenarios with Random embeddings. This phenomenon also occurred in the experiment labeled ✗ EL2 (ID: 17), which stands out as the worst result among all experiments involving BERT embeddings. Furthermore, all these scenarios share the characteristic of $\mu = 0.1$ and an MLP with few neurons. Additionally, a marked bar indicates the EL2 stage in all four cases, meaning a second refinement was performed on an already refined embedding. These factors, particularly within the context of this dataset, contributed to the classifier's poor performance. It is worth noting that class imbalance is even more pronounced in this dataset compared to the Laptops dataset, as illustrated in Table 1.

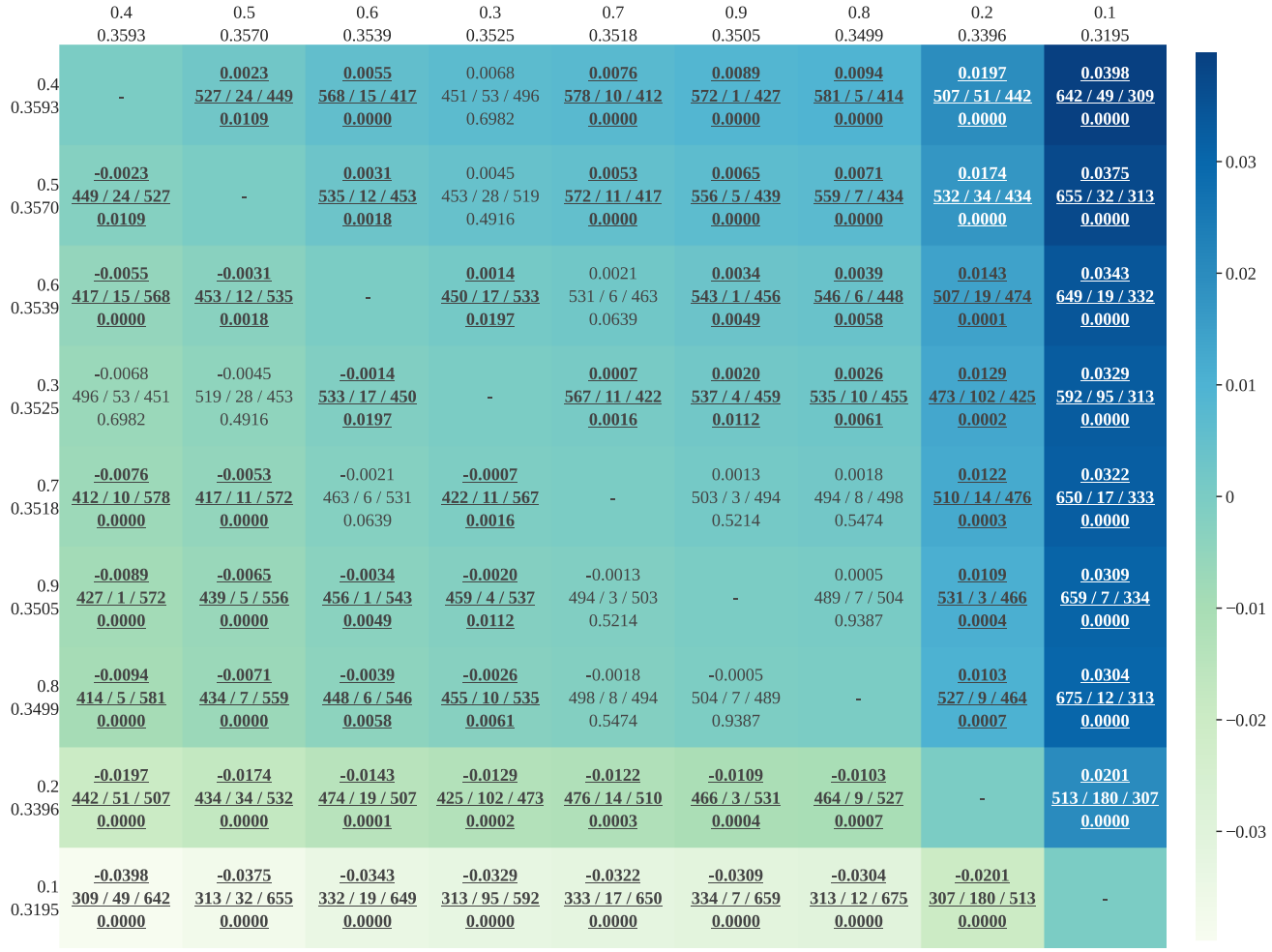

Fig. 8 MCM for μ with Random embeddings

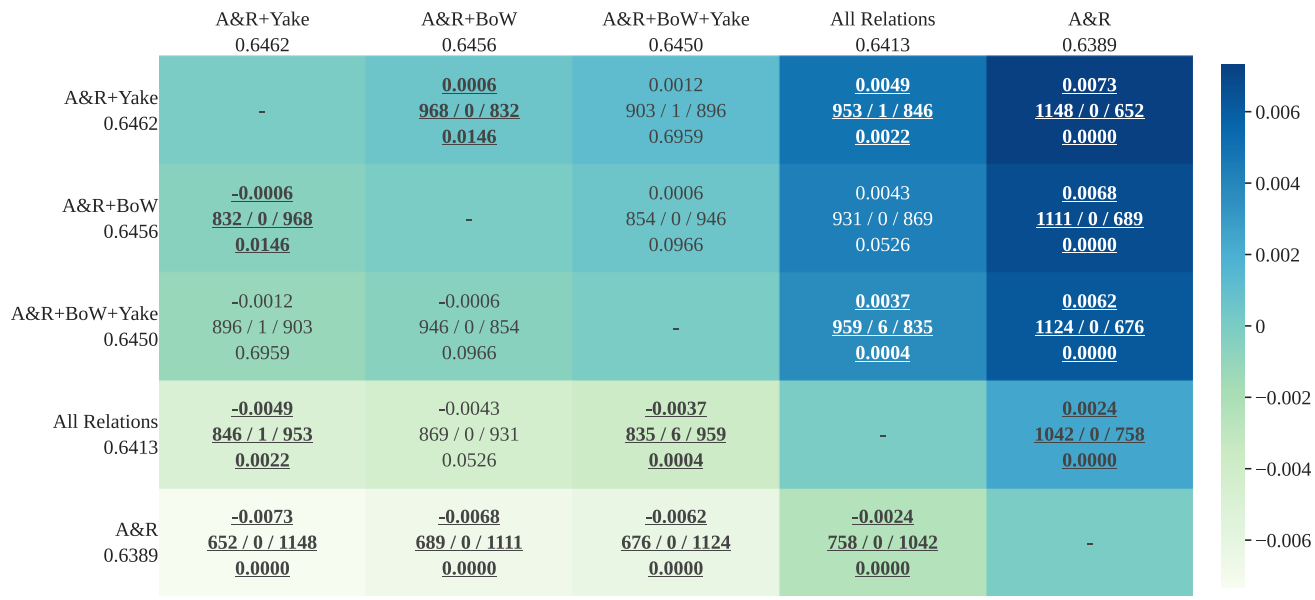

Fig. 9 MCM for μ with BERT embeddings

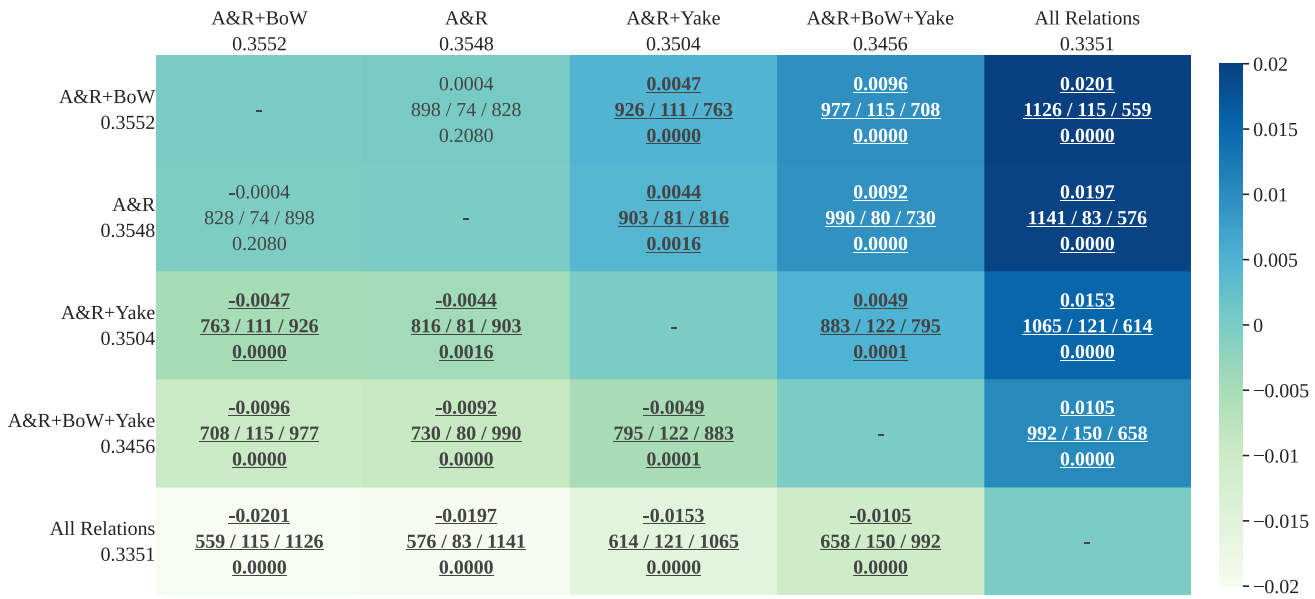

Fig. 10 MCM for μ with Random embeddings

Table 4 Details of settings chosen for detailed analysis. The "Details" column indicates whether it was chosen because it was the best (✓) or worst (✗) F1-Macro average

ID	Dataset	Embedding	Classifier	μ	Graph	Details
1	Laptops	bert	MLP(100)	0.1	A&R	✗ Statistical
2	Laptops	bert	MLP(768)	0.3	A&R+Yake	✓ Statistical
3	Laptops	bert	MLP(768)	0.7	A&R+BoW	✗ E1
4	Laptops	bert	MLP(768)	0.1	A&R+BoW	✓ E1
5	Laptops	bert	MLP(128)	0.1	All Relations	✗ EL2
6	Laptops	bert	MLP(768)	0.3	A&R+BoW	✓ EL2
7	Laptops	random	MLP(100)	0.1	All Relations	✗ Statistical
8	Laptops	random	MLP(768)	0.4	A&R+BoW	✓ Statistical
9	Laptops	random	MLP(100)	0.6	A&R+BoW	✗ E1
10	Laptops	random	MLP(768)	0.8	A&R	✓ E1
11	Laptops	random	MLP(128)	0.1	All Relations	✗ EL2
12	Laptops	random	MLP(768)	0.2	A&R+BoW	✓ EL2
13	Restaurants	bert	MLP(100)	0.1	A&R	✗ Statistical
14	Restaurants	bert	MLP(768)	0.3	A&R+Yake	✓ Statistical
15	Restaurants	bert	MLP(256)	0.2	A&R+BoW	✗ E1
16	Restaurants	bert	MLP(768)	0.8	A&R	✓ E1
17	Restaurants	bert	MLP(100)	0.1	All Relations	✗ EL2
18	Restaurants	bert	MLP(512)	0.3	All Relations	✓ EL2
19	Restaurants	random	MLP(100)	0.1	All Relations	✗ Statistical
20	Restaurants	random	MLP(768)	0.4	A&R+BoW	✓ Statistical
21	Restaurants	random	MLP(100)	0.1	A&R+Yake	✗ E1
22	Restaurants	random	MLP(768)	0.3	A&R	✓ E1
23	Restaurants	random	MLP(100)	0.1	A&R	✗ EL2
24	Restaurants	random	MLP(256)	0.3	A&R+BoW	✓ EL2

Hence, this occurrence was more frequent in this dataset. Conversely, when examining the best results overall, most are above the line representing the mean (Baseline/Strong Baseline), with some coming close to the line indicating the best result (Baseline/Strong Baseline). In contrast, for Random embeddings, the best results surpass the baseline.

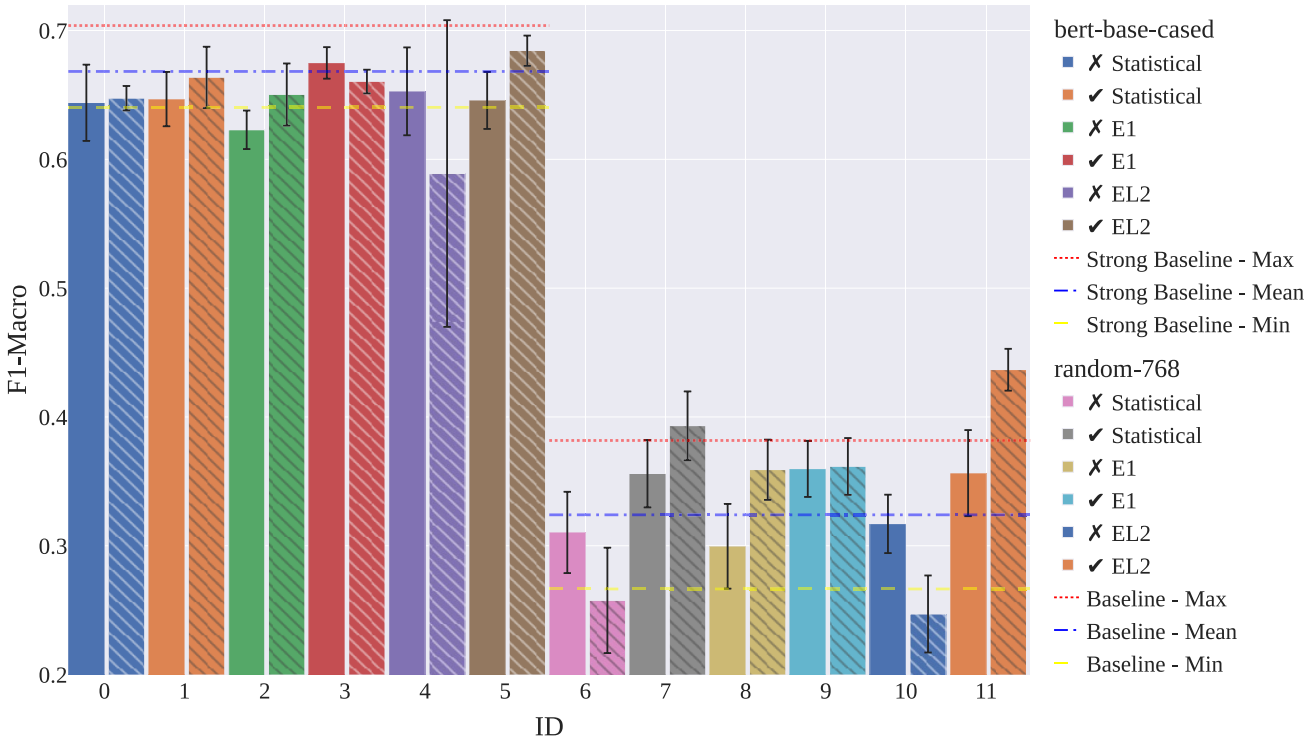

Fig. 11 Results obtained for the dataset Laptop. The x-axis (ID) indicates the item in Table 4. The error bars illustrate the standard deviation

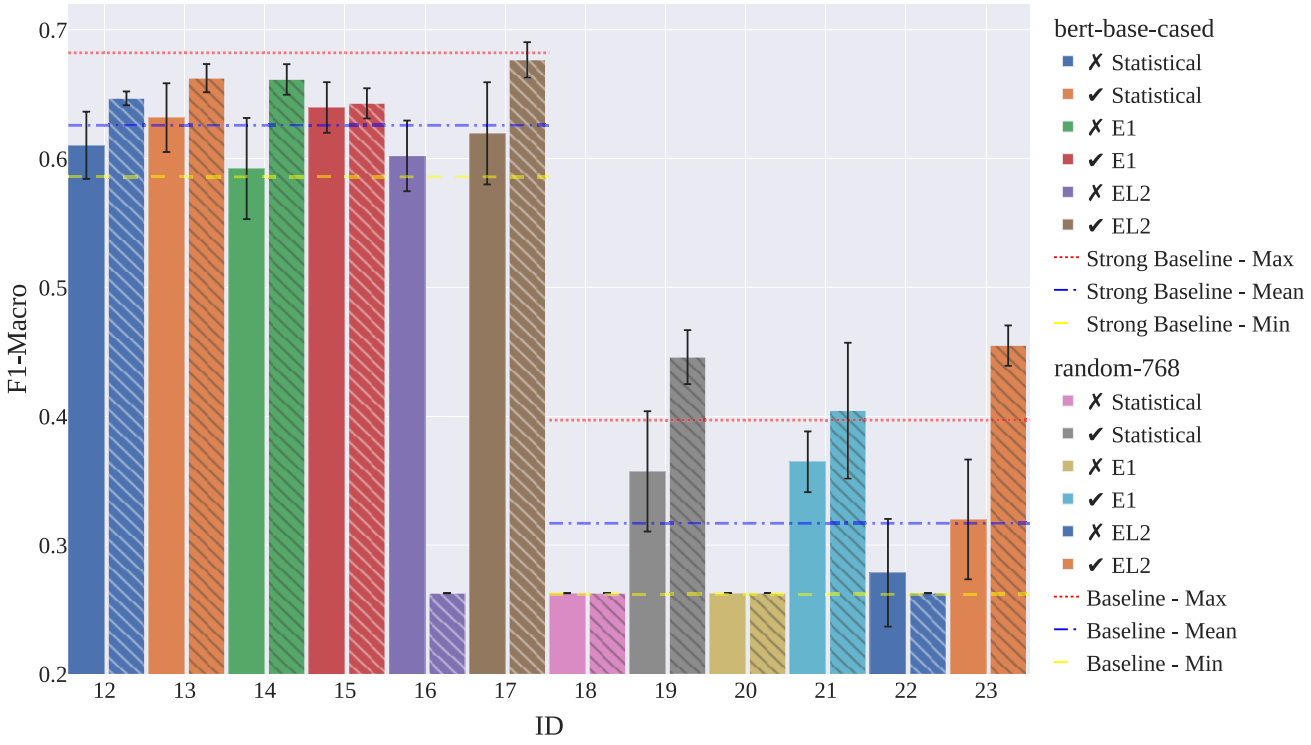

Fig. 12 Results obtained for the dataset Restaurants. The x-axis (ID) indicates the item in Table 4. The error bars illustrate the standard deviation

All the experiments presented in this section demonstrate that it is possible to refine embeddings using the proposed approach while remaining competitive with the Strong Baseline/Baseline. Although computational cost has not been explored, it is common for label propagation to be faster than fine-tuning a language model. Additionally, in the case of huge graphs, graph sampling is feasible, as described by Santos et al. [59]. Another advantage is that, after applying our proposed approach, all vertices have embeddings, enabling their use in other tasks. We will delve into this aspect in Section 4.6.

4.6 Embedding analysis

In the previous section, we evaluated the embeddings in the target task and compared them to a language model. This section will qualitatively analyze the embeddings and utilize other vertices from the heterogeneous graph to illustrate potential applications.

This analysis aims to demonstrate that the embeddings have captured not only semantic information but also the graph's structure. We will perform queries involving algebraic operations with embeddings of specific vertices to achieve this. Using the resulting embedding, we will identify the ten most similar vertices between the Aspect and Yake layers.

We will use the "A&R+Yake" subgraph and $\mu = 0.3$, which yielded the best statistical results with BERT, as shown in Table 4. Additionally, we will assess Random embeddings, in this case using $\mu = 0.4$, which resulted in the best statistical performance. However, we will maintain the subgraph "A&R+Yake" to ensure the presence of the Yake layer.

For comparison purposes, we will use BERT without fine-tuning as the baseline, and BERT fine-tuned at the optimal epoch for each dataset as the Strong Baseline. To do this, we will generate embeddings (Section 4.4) for each of these models based on Equation 6 for both aspects and Yake keywords.

Two types of queries were conducted: the first involved adding the embeddings of two vertices, and the second sum the embeddings of two vertices and subtracting using embedding another vertex.

A heatmap was created to illustrate the results and display the retrieved vertices. Each cell in the heatmap consists of two values: the first indicates the retrieved vertex in the format "vertex name [layer]", and the second indicates the pertinence of sentiments normalized through "sum normalization". The format displayed in the figure is "negative/neutral/positive".

The color represents the weighted average of sentiments, based on Equation 7, which calculates the weighted average for a given vertex o_i. In this context, the closer the color is to 1, the more it indicates a positive sentiment; the closer it is to 0 indicates a neutral sentiment, and finally, the closer it is to -1 indicates a negative sentiment.

$$M = -1 \times \mathcal{L}_{o_i,\text{negative}} + 0 \times \mathcal{L}_{o_i,\text{neutral}} + 1 \times \mathcal{L}_{o_i,\text{positive}} \tag{7}$$

The sentiment for each vertex is derived from the label propagation performed in Section 4.4, for the "A&R+Yake" subgraph using BERT (results illustrated in Fig. 4). Furthermore, we used the same sentiment source for all retrieved vertices in the figure, regardless of which embedding recovered each vertex.

The results for the Laptops dataset are illustrated in Figures 13 and 14, while the results for the Restaurants dataset are shown in Figures 15 and 16. It is noticeable that, overall, both the baseline and the Strong Baseline models provided vertices representing synonyms and words in the same context, as expected from language models. In contrast, our approach yielded a wider variety of vertices, many related to the quality of service or the item used for the query. This becomes more evident, especially in the Restaurants dataset; for example, in Figure 15, the query involves "service" and "staff", and upon analyzing the retrieved vertices, we find phrases like "extremely friendly", "excellent service", and "staff is friendly", which are characteristics of the staff and the service quality of the establishment.

These results demonstrate that the embeddings have also captured the topology of the heterogeneous graph and incorporated other contexts, not just semantically similar vertices. This characteristic allows these embeddings to be utilized in various tasks, simultaneously serving in NLP tasks such as aspect-based sentiment analysis.

5 Conclusion

This article explored the importance of embeddings in heterogeneous graphs and the challenges that arise due to the inherent diversity of these graphs, which can contain various vertex and edge types. Throughout the work, we discussed how these vector representations are used for various applications, from classification to recommendation and similarity search. Additionally, we highlighted the relevance of language model embeddings, such as BERT, in this context, leveraging their ability to capture semantic and contextual information in text associated with vertices. This work proposed an approach that combines label propagation techniques with language models to enhance embeddings in heterogeneous graphs, focusing on the quality of vector representations, considering both the graph's structure and the semantic richness of textual data and task class information.

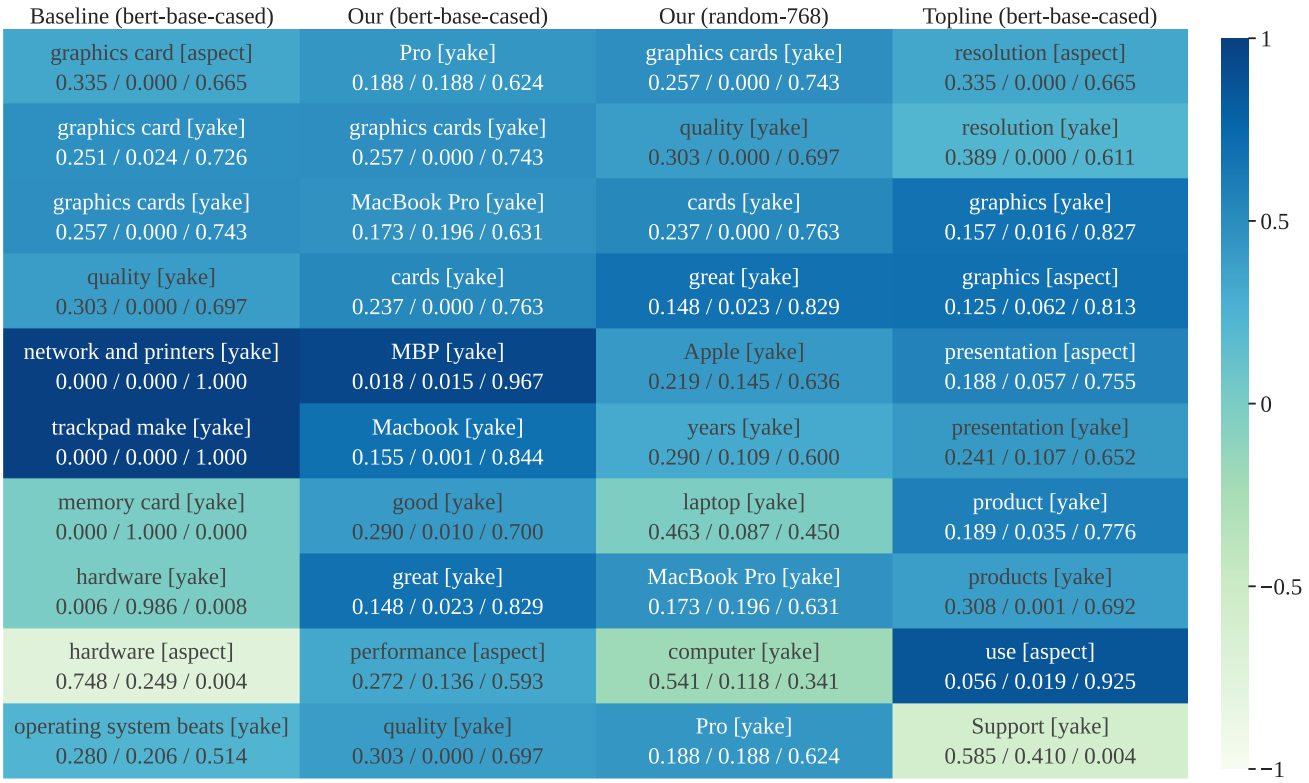

Fig. 13 Results obtained for the Laptops dataset using the query "graphics cards [aspect] + quality [aspect]". The order of vertices in the column indicates the similarity between the vertex embedding and the query. The word in brackets indicates the layer of vertex that was retrieved

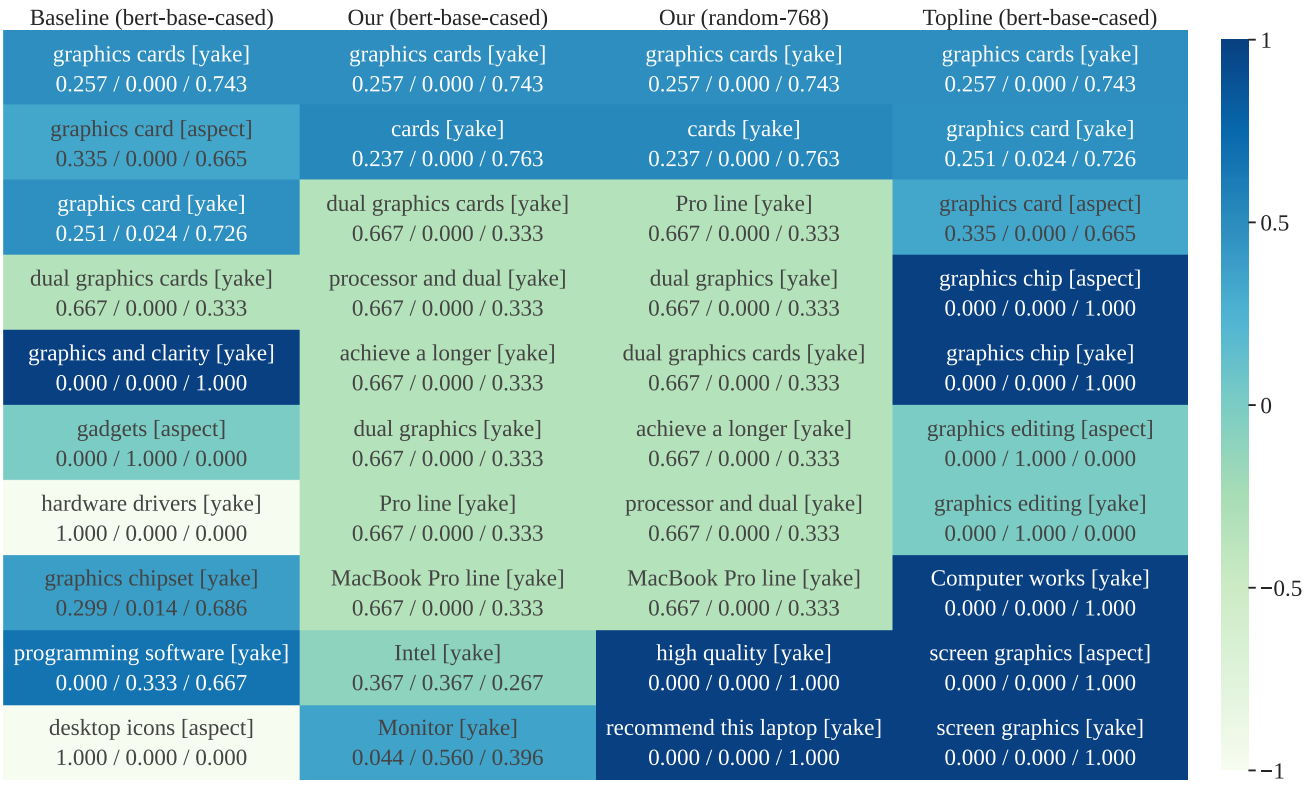

Fig. 14 Results obtained for the Laptops dataset using the query "graphics cards [aspect] + quality [aspect] - price [aspect]". The order of vertices in the column indicates the similarity between the vertex embedding and the query. The word in brackets indicates the layer of vertex that was retrieved

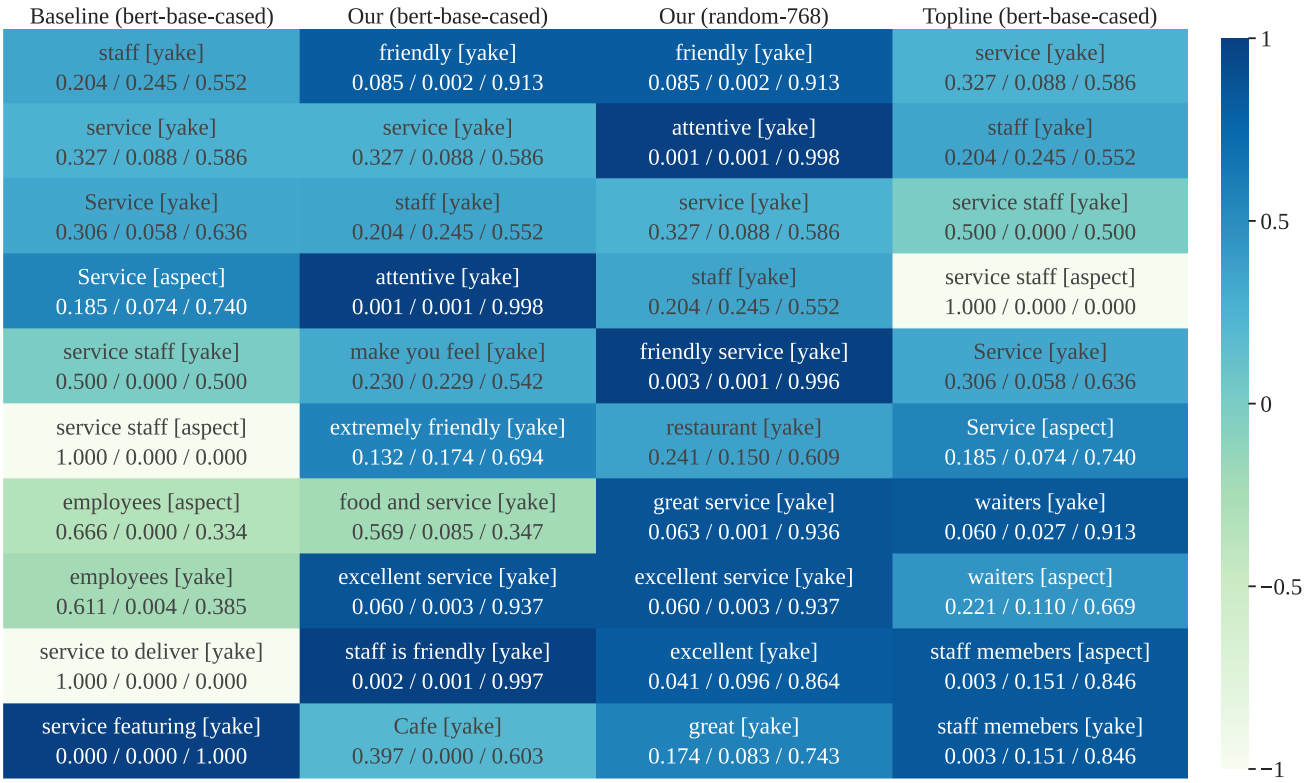

Fig. 15 Results obtained for the Restaurants dataset using the query "service [aspect] + staff [aspect]". The order of vertices in the column indicates the similarity between the vertex embedding and the query. The word in brackets indicates the type of layer that was retrieved

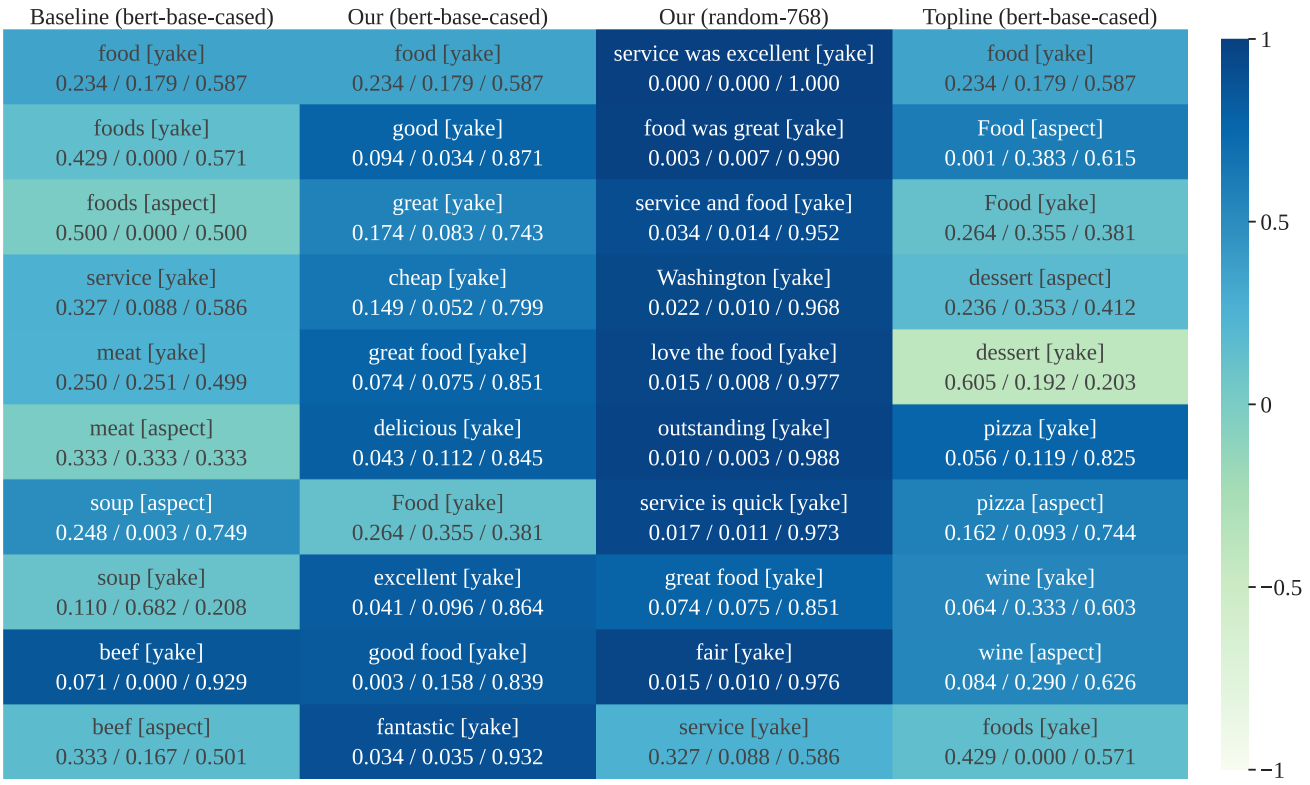

Fig. 16 Results obtained for the Restaurants dataset using the query "food [aspect] + service [aspect] - staff [aspect]". The order of vertices in the column indicates the similarity between the vertex embedding and the query. The word in brackets indicates the layer of vertex that was retrieved

We conducted a comparative analysis between the proposed approach and fine-tuned BERT on an aspect-based sentiment analysis task. The results were promising, as our method showed competitive results, approaching the average performance of the language model and, in some cases, even slightly outperforming it. This highlights the effectiveness of our approach in enhancing vector representations in heterogeneous graphs, especially when applied to complex aspect-based sentiment analysis tasks.

Furthermore, this article also explored some possibilities for applying the embeddings of auxiliary vertices, highlighting one of the key differences between our approach and the exclusive use of a language model. This versatility underscores the potential of the embeddings generated by our method in different contexts and applications, expanding their usefulness in various data analysis and natural language processing scenarios.

In other words, our approach fine-tuned the initial embeddings through the heterogeneous graph while propagating this information throughout the graph. This process allows all vertices in the graph to have an embedding that represents the graph's structure and the contextual knowledge from the language model's embedding. Another advantage of the proposed approach is its lightweight nature, enabling it to run on CPUs, while many language models require GPUs. It is worth noting that although this approach has been evaluated in the transductive scenario, it can also be applied to the inductive scenario. To do so, add the new vertices to the graph and continue training for these vertices or the entire graph.

Section 3 of this article presented an approach divided into two main stages: (1) the initial generation of embeddings (Section 3.1) and labels (Section 3.2) for all graph vertices and (2) the refinement of these vector representations through label propagation within a regularization framework (Section 3.3). Sections 3.1 and 3.2 suggest alternative methods for embedding generation, label assignment, and propagation strategies. Based on this, future work could investigate variations in the initial embedding techniques, experiment with different propagation algorithms, and explore diverse heterogeneous graph architectures. Furthermore, regarding hyperparameter selection, future studies may use the values from the "✓ Statistical" configuration listed in Table 4 as a starting point since this setup proved consistent and may serve as a helpful baseline for other datasets and graph structures.

One of the limitations of the current approach is related to the stopping condition during the propagation of embeddings. Our method currently uses convergence, which checks how much the embeddings have changed from one iteration to another. If this difference is smaller than a threshold, the algorithm stops propagation. While we have achieved good results with this approach, it does not guarantee that we have extracted the maximum potential, as performing many iterations can make the vertex embeddings very close together, which may not be ideal depending on the use of these embeddings. Therefore, future work includes searching for a better way to stop the propagation of embeddings.

Finally, other directions for future work include exploring the embeddings generated by the proposed method in other scenarios, such as recommendation systems, community detection, and link prediction. Testing the proposed method in an inductive scenario, evaluating how to deal with new vertices and edges after the propagation of embeddings has finished.

Beyond the specific contributions described above, this work will also serve other researchers who study heterogeneous graphs. By integrating label propagation with language model embeddings, our approach provides a lightweight and versatile framework that can be used as a baseline for exploring new architectures, propagation strategies, and embedding techniques. This combination bridges structural and semantic information, which is a central challenge in heterogeneous graph research. Moreover, the integration of language models and word embeddings opens new frontiers for learning with heterogeneous networks, since future studies may exploit large language models not only to enrich vertex embeddings, but also to generate pseudolabels and even refine graph connectivity by suggesting new edges. We expect that future studies build upon this idea to design more efficient, interpretable, and domain-adapted methods, thereby advancing the broader field of heterogeneous graph representation learning.

6 Supplementary information

Not applicable

Acknowledgements This study was financed in part by the Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) - Process 2019/25010-5, 2022/09091-8 and 2023/10100-4. Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) - Process 309575/2021-4, 316507/2023-7 and 307184/2025-0. Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. We also thank the National Institute of Artificial Intelligence (IAIA) of the CNPq. This work is funded by national funds through FCT – Fundação para a Ciência e a Tecnologia, I.P., under the support UID/50014/2023 (https://doi.org/10.54499/UID/50014/2023). A. Jorge and R. Campos supported by the Portuguese Foundation for Science and Technology (FCT) under the support of the StorySense project (DOI 10.54499/2022.09312.PTDC).

Funding The Article Processing Charge (APC) for the publication of this research was funded by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) (ROR identifier: 00x0ma614).

Data Availability The data used is available at https://alt.qcri.org/semeval2014/task4/index.php?id=data-and-tools.

Code Availability The code that supports the findings of this study is openly available on GitHub at https://github.com/BrucceNeves/EPHG-CR.

Declarations

Conflict of Interests The authors declare no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Shi C, Li Y, Zhang J, Sun Y, Philip SY (2016) A survey of heterogeneous information network analysis. IEEE Trans Knowl Data Eng 29(1):17–37
2. Matsuno IP, Rossi RG, Marcacini RM, Rezende SO (2016) Aspect-based sentiment analysis using semi-supervised learning in bipartite heterogeneous networks. J Inf Data Manage 7(2):141
3. Arruda HF, Silva FN, Costa LdF, Amancio DR (2017) Knowledge acquisition: A complex networks approach. Inf Sci 421:154–166
4. Shi C, Philip SY (2017) Heterogeneous Information Network Analysis and Applications. Data Analytics. Springer, New York, NY
5. Marcacini RM, Rossi RG, Matsuno IP, Rezende SO (2018) Cross-domain aspect extraction for sentiment analysis: A transductive learning approach. Decis Support Syst 114:70–80
6. Carneiro MG, Cheng R, Zhao L, Jin Y (2019) Particle swarm optimization for network-based data classification. Neural Netw 110:243–255
7. Xia F, Sun K, Yu S, Aziz A, Wan L, Pan S, Liu H (2021) Graph learning: A survey. IEEE Trans Artif Intell 2(2):109–127
8. Liu J, Shi C, Yang C, Lu Z, Philip SY (2022) A survey on heterogeneous information network based recommender systems: Concepts, methods, applications and resources. AI Open 3:40–57
9. Wang X, Bo D, Shi C, Fan S, Ye Y, Philip SY (2022) A survey on heterogeneous graph embedding: methods, techniques, applications and sources. IEEE Trans Big Data 9(2):415–436
10. Yang C, Xiao Y, Zhang Y, Sun Y, Han J (2020) Heterogeneous network representation learning: A unified framework with survey and benchmark. IEEE Trans Knowl Data Eng 34(10):4854–4873
11. Bing R, Yuan G, Zhu M, Meng F, Ma H, Qiao S (2023) Heterogeneous graph neural networks analysis: a survey of techniques, evaluations and applications. Artif Intell Rev 56(8):8003–8042
12. Fu T-y, Lee W-C, Lei Z (2017) Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp 1797–1806
13. Xu L, Wei X, Cao J, Yu PS (2017) Embedding of embedding (eoe) joint embedding for coupled heterogeneous networks. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, pp 741–749
14. Hu B, Fang Y, Shi C (2019) Adversarial learning on heterogeneous information networks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp 120–129
15. Fu X, Zhang J, Meng Z, King I (2020) Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In: Proceedings of the Web Conference 2020, pp 2331–2341
16. Hong H, Guo H, Lin Y, Yang X, Li Z, Ye J (2020) An attention-based graph neural network for heterogeneous structural learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol 34, pp 4132–4139
17. Xue H, Yang L, Jiang W, Wei Y, Hu Y, Lin Y (2021) Modeling dynamic heterogeneous network for link prediction using hierarchical attention with temporal rnn. In: Proceedings of the Machine Learning and Knowledge Discovery in Databases, pp 282–298
18. Bengio Y, Courville A, Vincent P (2013) Representation learning: A review and new perspectives. IEEE Trans Pattern Anal Mach Intell 35(8):1798–1828
19. Guo W, Wang J, Wang S (2019) Deep multimodal representation learning: A survey. IEEE Access 7:63373–63394
20. Joloudari JH, Hussain S, Nematollahi MA, Bagheri R, Fazl F, Alizadehsani R, Lashgari R, Talukder A (2023) Bert-deep cnn: state of the art for sentiment analysis of covid-19 tweets. Soc Netw Anal Min 13(1):99
21. Yang KK, Wu Z, Bedbrook CN, Arnold FH (2018) Learned protein embeddings for machine learning. Bioinformatics 34(15):2642–2648
22. Kalyan KS, Sangeetha S (2020) Secnlp: A survey of embeddings in clinical natural language processing. J Biomed Inform 101:103323
23. Patil R, Boit S, Gudivada V, Nandigam J (2023) A survey of text representation and embedding techniques in nlp. IEEE Access
24. Chen H, Yin H, Wang W, Wang H, Nguyen QVH, Li X (2018) Pme: projected metric embedding on heterogeneous networks for link prediction. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp 1177–1186
25. Dong Y, Chawla NV, Swami A (2017) metapath2vec: Scalable representation learning for heterogeneous networks. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 135–144
26. Zhang W, Fang Y, Liu Z, Wu M, Zhang X (2020) mg2vec: Learning relationship-preserving heterogeneous graph representations via metagraph embedding. IEEE Trans Knowl Data Eng 34(3):1317–1329
27. Wang X, Ji H, Shi C, Wang B, Ye Y, Cui P, Yu PS (2019) Heterogeneous graph attention network. In: Proceedings of the The World Wide Web Conference, pp 2022–2032
28. Wang X, Lu Y, Shi C, Wang R, Cui P, Mou S (2020) Dynamic heterogeneous information network embedding with meta-path based proximity. IEEE Trans Knowl Data Eng 34(3):1117–1132
29. Belkin M, Matveeva I, Niyogi P (2004) Regularization and semi-supervised learning on large graphs. In: Proceedings of the 17th Conference on Learning Theory, pp 624–638
30. Ji M, Sun Y, Danilevsky M, Han J, Gao J (2010) Graph regularized transductive classification on heterogeneous information networks. In: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases, pp 570–586
31. Delalleau O, Bengio Y, Le Roux N (2005) Efficient non-parametric function induction in semi-supervised learning. In: Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics, pp 96–103

32. Garcia Duran A, Niepert M (2017) Learning graph representations with embedding propagation. In: Advances in Neural Information Processing Systems, pp 5119–5130
33. Zheng VW, Sha M, Li Y, Yang H, Fang Y, Zhang Z, Tan K-L, Chang KC-C (2018) Heterogeneous embedding propagation for large-scale e-commerce user alignment. In: Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM), pp 1434–1439
34. Yang C, Liu L, Liu M, Wang Z, Zhang C, Han J (2020) Graph clustering with embedding propagation. In: Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), pp 858–867
35. Carmo P, Marcacini R (2021) Embedding propagation over heterogeneous event networks for link prediction. In: Proceedings of the 2021 IEEE International Conference on Big Data (Big Data), pp 4812–4821
36. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 701–710
37. Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 855–864
38. Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, vol 30, pp 1024–1034
39. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: Proceedings of the International Conference on Learning Representations
40. Devlin J, Chang M-W, Lee K, Toutanova K (2019) BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp 4171–4186
41. Tenney I, Xia P, Chen B, Wang A, Poliak A, McCoy RT, Kim N, Durme BV, Bowman S, Das D, Pavlick E (2019) What do you learn from context? probing for sentence structure in contextualized word representations. In: International Conference on Learning Representations, pp 1–17
42. Hewitt J, Manning CD (2019) A structural probe for finding syntax in word representations. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1, pp 4129–4138
43. Jawahar G, Sagot B, Seddah D (2019) What does BERT learn about the structure of language? In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp 3651–3657
44. Dos Santos BN, Marcacini RM, Rezende SO (2021) Multi-domain aspect extraction using bidirectional encoder representations from transformers. IEEE Access 9:91604–91613
45. Lu Q, Getoor L (2003) Link-based classification. In: Proceedings of the 20th International Conference on Machine Learning, pp 496–503
46. Hensman S (2004) Construction of conceptual graph representation of texts. In: Proceedings of the Student Research Workshop at HLT-NAACL 2004, pp 49–54
47. Gee KR, Cook DJ (2005) Text classification using graph-encoded linguistic elements. In: Proceedings of the 18th Florida Artificial Intelligence Research Society Conference, pp 487–492
48. Solé RV, Corominas-Murtra B, Valverde S, Steels L (2010) Language networks: their structure, function, and evolution. Complexity 15(6):20–26
49. Mishra M, Huan J, Bleik S, Song M (2012) Biomedical text categorization with concept graph representations using a controlled vocabulary. In: Proceedings of the 11th International Workshop on Data Mining in Bioinformatics, pp 26–32
50. Aggarwal C, Zhao P (2013) Towards graphical models for text processing. Knowl Inf Syst 36(1):1–21
51. Rossi RG, Andrade Lopes A, Paulo Faleiros T, Rezende SO (2014) Inductive model generation for text classification using a bipartite heterogeneous network. J Comput Sci Technol 3(29):361–375
52. Rossi RG, Andrade Lopes A, Rezende SO (2016) Optimization and label propagation in bipartite heterogeneous networks to improve transductive classification of texts. Inf Process Manage 52(2):217–257
53. Santos BN, Rossi RG, Rezende SO, Marcacini RM (2020) A two-stage regularization framework for heterogeneous event networks. Pattern Recogn Lett 138:490–496
54. Zhu X, Ghahramani Z, Lafferty J (2003) Semi-supervised learning using gaussian fields and harmonic functions. In: Proceedings of the 20th International Conference on Machine Learning, pp 912–919
55. Zhou D, Bousquet O, Lal TN, Weston J, Schölkopf B (2004) Learning with local and global consistency. In: Proceedings of the Advances in Neural Information Processing Systems (NIPS), vol 16, pp 321–328
56. Osaku D, Santos PM, Santos BN, Rezende SO (2023) Pasture degradation papers search: how can supervised and transductive methods help on the process of classification? In: Proceedings of the XX Encontro Nacional de Inteligência Artificial e Computacional, pp 1–15
57. Zhu X (2005) Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison
58. Campos R, Mangaravite V, Pasquali A, Jorge A, Nunes C, Jatowt A (2020) Yake! keyword extraction from single documents using multiple local features. Inf Sci 509:257–289
59. Santos BN, Marcacini RM, Rezende SO (2019) A sampling-based framework for transductive classification in information networks. In: Proceedings of the 8th Brazilian Conference on Intelligent Systems (BRACIS), pp 657–662
60. Ismail-Fawaz A, Dempster A, Tan CW, Herrmann M, Miller L, Schmidt DF, Berretti S, Weber J, Devanne M, Forestier G, et al (2023) An approach to multiple comparison benchmark evaluations that is stable under manipulation of the comparate set. arXiv:2305.11921
61. Wilcoxon F (1992) Individual comparisons by ranking methods. In: Proceedings of the Breakthroughs in Statistics: Methodology and Distribution, pp 196–202