RELATIONAL REPRESENTATION LEARNING INCORPORATING TEXTUAL COMMUNICATION FOR SOCIAL NETWORKS

by

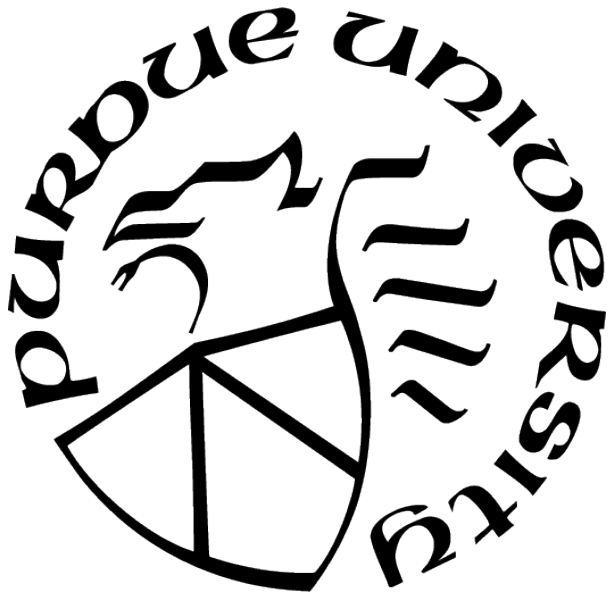
Yi-Yu Lai

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy



Department of Computer Science

West Lafayette, Indiana

May 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Jennifer Neville, Chair

Department of Computer Science

Dr. Dan Goldwasser

Department of Computer Science

Dr. Ninghui Li

Department of Computer Science

Dr. Clifton W. Bingham

Department of Computer Science

Approved by:

Dr. Kihong Park

To my beloved family.

ACKNOWLEDGMENTS

Throughout the writing of this dissertation, I have received a great deal of support and assistance from many people.

First of all, I would like to express my sincere gratitude to my advisor, Professor Jennifer Neville, who supported me both academically and financially throughout my Ph.D. study with invaluable expertise in formulating research questions and methodology with creative and critical thinking. Her insightful guidance and feedback pushed me to sharpen my skills and brought my work to a higher level. I am also very grateful to Professor Dan Goldwasser, Professor Ninghui Li, and Professor Chris Clifton as being my Ph.D. committee members for their thoughtful advice and comments that improved my dissertation and their encouragement over the years.

I want to thank all my lab mates and collaborators (in alphabetical order): Nesreen Ahmed, Iman Alodah, Sait Celebi, Timothy Fond, Mahak Goindani, Guilherme Gomes, Mengyue Hang, Suvidha Kancharla, Praveen Kumar, Chang Li, Ying-Chun Lin, Jason Meng, John Moore, Sebastian Moreno, Hogun Park, Joel Pfeiffer, Pablo Robles-Granda, Susheel Suresh, Xi Tan, Jiasen Yang, Giselle Zeno, and Shandian Zhe. I sincerely appreciate all the discussions that we had. We exchanged knowledge and ideas from different perspectives of our domains, in which we learned from each other and worked towards our goals.

Special thanks to Dr. Anthony J.T. Lee, Dr. Hsun-Ping Hsieh, and Shih-Feng Yang. Thank you for supporting and encouraging me to pursue my Ph.D. study.

I feel so thankful to all the friends I met at Purdue for their assistance. The times we have spent together with laughter become my precious memories.

Lastly, I would like to show my genuine appreciation to my family for their endless love and encouragement. Thanks to my parents for their wise counsel, sympathetic ears, and long-lasting support. You are always there for me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Representation learning (RL) for social networks facilitates real-world tasks such as visualization, link prediction and friend recommendation. Many methods have been proposed in this area to learn continuous low-dimensional embedding of nodes, edges or relations in social and information networks. However, most previous network RL methods neglect *social signals*, such as textual communication between users (nodes). Unlike more typical binary features on edges, such as post likes and retweet actions, social signals are more varied and contain ambiguous information. This makes it more challenging to incorporate them into RL methods, but the ability to quantify social signals should allow RL methods to better capture the implicit relationships among real people in social networks. Second, most previous work in network RL has focused on learning from homogeneous networks (i.e., single type of node, edge, role, and direction) and thus, most existing RL methods cannot capture the *heterogeneous* nature of relationships in social networks. Based on these identified gaps, this thesis aims to study the feasibility of incorporating heterogeneous information, e.g., texts, attributes, multiple relations and edge types (directions), to learn more accurate, fine-grained network representations.

In this dissertation, we discuss a preliminary study and outline three major works that aim to incorporate textual interactions to improve relational representation learning. The preliminary study learns a joint representation that captures the textual similarity in content between interacting nodes. The promising results motivate us to pursue broader research on using social signals for representation learning. The first major component aims to learn explicit node and relation embeddings in social networks. Traditional knowledge graph (KG) completion models learn latent representations of entities and relations by interpreting them as translations operating on the embedding of the entities. However, existing approaches do not consider textual communications between users, which contain valuable information to provide meaning and context for social relationships. We propose a novel approach that incorporates textual interactions between each pair of users to improve representation learning of both users and relationships. The second major component focuses on analyzing how users interact with each other via natural language content. Although the data is inter-

connected and dependent, previous research has primarily focused on modeling the social network behavior separately from the textual content. In this work, we model the data in a holistic way, taking into account the connections between the social behavior of users and the content generated when they interact, by learning a joint embedding over user characteristics and user language. In the third major component, we consider the task of learning edge representations in social networks. Edge representations are especially beneficial as we need to describe or explain the relationships, activities, and interactions among users. However, previous work in this area lack well-defined edge representations and ignore the relational signals over multiple views of social networks, which typically contain multi-view contexts (due to multiple edge types) that need to be considered when learning the representation. We propose a new methodology that captures asymmetry in multiple views by learning well-defined edge representations and incorporates textual communications to identify multiple sources of social signals that moderate the impact of different views between users.

1. INTRODUCTION

Representation learning on large-scale networks has attracted a lot of attention recently because of the methods' ability to convert a sparse and high-dimensional network structure into a low-dimensional network embedding [1]. Learned network embeddings can be treated as features and used on various prediction and analysis tasks (e.g., node classification, link prediction and node visualization). For example, we can predict the link or relationship by distance or similarity of two node embeddings. Many network representation learning methods [2]–[5] have been proposed to learn continuous embedding vectors for nodes and demonstrated to be effective in preserving the structure of complex networks.

More specifically, network embedding methods typically consider a network graph defined as $G = (V, E)$ where V is a set of nodes and E is a set of edges between the nodes. Each edge $\mathrm{e} \in E$ is an ordered pair of two nodes, $\mathrm{e}_{uv} := (u, v)$ where u, $v \in V$. When applied to *social network data* with attributes and textual interactions, the given network graph is extended to $G = (V, E, A, D)$ where V represents users, E represents relationships between users, A is a set of user attributes associated with each user and D is a collection of exchanged messages between the users. User attribute $a_u \in A$ is collected from user u's profile and group memberships. Document $d_{uv} \in D$ represents the set of posts t_{uv} sent from user u to user v. A set of relations $R = \{r_k\}$ over pairs of users are either defined by the attribute values that they share in common (e.g., $r_k(u, v) = 1$ if $a_u = a_v = k$ and 0 otherwise), or defined by certain directional attributes (e.g., v is u's top friend, v is senior to u). The task of network embedding involves learning a low-dimensional vector representation $x_u \in \mathbb{R}^d$ for every node $u \in V$ (where $d \ll |V|$) that preserves proximities between nodes, learning an edge representation $y_{uv} \in \mathbb{R}^{2d}$ for every edge e_{uv} that reflects the relationships from u to v, and/or learning a relation representation $z_r \in \mathbb{R}^d$ for every relation type $r \in R$.

We note that previous studies of network embedding learning have primarily focused on coarse network structure, and as such failed to preserve many of the more fine-grained characteristics of social networks. In particular, there are several deficiencies when applying traditional network representation learning approaches to social networks:

1. Nodes in a social network represent users. A user usually has multiple roles regarding different relationships with other users. Representing a user by a single node embedding is not sufficient to capture the user's traits comprehensively. Also, a user should be represented differently based on whether he/she is the source or the destination of a directed relation with another user [6].

2. Social network data often contain rich *textual communication* among users and this information is a valuable signal about the types and strength of relationships between users. However, to date, textual information has not been used effectively in network embedding methods.

3. Network embedding works mainly focus on learning latent representation for nodes (users) and there is a lack of well-defined edge representation used in previous RL works. Node embeddings are limited in their ability to represent the variety of relationships between users. Moreover, there is no principled theoretical framework to represent edges through node embeddings directly.

To address the first deficiency, it is feasible to adopt knowledge graph (KG) completion approaches [7]–[10] where a KG consists of facts stored in the form of triplet: (head entity, relation, tail entity). Knowledge graph embedding aims to embed components of a KG including entities and relations into continuous and low-dimensional vector spaces, so as to simplify their manipulation while preserving the inherent structure of the KG [11]. When applying KG-based models to social networks, users are represented as entities, with multiple relations among them. We note that given a pair of users and their relation as a triplet (u, r, v), the goal is to learn a joint embedding for users and relations, such that every relation can be viewed as a translation of users in the embedding space. Let x_u and x_v denote the user embeddings of u and v, and z_r denote the embedding of relation r. The goal then is to make the embedding x_u close to that of x_v by adding the relationship embedding z_r (i.e., $x_u + z_r \approx x_v$). The embeddings of users and relations are in the same space $\in \mathbb{R}^d$.

In this context, semantic content of entities can also provide abundant information to improve the learning process. [12], [13] take the description of entities into account when learning embeddings. However, they do not incorporate the textual communication between

users, which provide more information about the strength and type of their relationships (as we describe in the second deficiency).

Some researchers have attempted to address the second deficiency, but these previous works fall into two, almost completely disconnected camps. The first focuses on social network analysis and looks at the network structure, and information flow on it, as means of inferring knowledge about the network. The other focuses on natural language analysis, looking into tasks such as extracting social relationships from narrative text [14]–[16] and analyzing the contents of the information flowing through the network. Both perspectives have produced a wide range of successful applications; however, they neglect to model the interactions between the social and linguistic representations and how they complement one another. A few recent exceptions look at the connections between the two camps. In [17], the network structure is represented explicitly and the authors use this to infer sentiment links between nodes in a social network, by combining local sentiment analysis decisions with global inference over the network structure. In [18], the network structure is represented by embedding users in a dense low dimensional space based on their social connections. In both of these works, there is a clear divide between social and language analysis. The social aspect is modeled separately and then used to improve the inferred predictions or disambiguate language use.

To address the third deficiency, and support the description and/or explanation of relationships, activities, or interactions between users, we need an effective framework for learning edge representations. Previous studies have consider edge representations that are simple functions of the two incident nodes' embeddings, i.e., through a computation of an average, concatenation [19], [20], or Hadamard product [21] (e.g. $x_u \oplus x_v$). There are some drawbacks to these approaches. First, they fail to consider the direction of edges. Second, they oversimplify the edge representation. In other words, these are very coarse ways to obtain an "edge representation", with little theoretical rationale for the approach. Although the entities and relations learned in knowledge graphs seem to be analogous to the node and edge embeddings in social networks, there is a fundamental difference. Every edge in a network is unique for a pair of nodes, while a relation embedding is a global embedding shared by all (head entity, tail entity) pairs who hold the relation. For example, given three

nodes u, v, s and two edges e_{uv}, e_{us} in a network, where e_{uv} represents a directed edge from source node u to destination node v. There exist two distinguishable edge representations y_{uv} and y_{us} in the latent embedding space. On the other hand, through knowledge graph based approaches, accordingly we have two triplets (u, r, v) and (u, r, s) by assuming a relation r associated to the edge in the network. It makes $x_u + z_r$ close to x_v, and $x_u + z_r$ close to x_s in the embedding space. This means the representations for edges y_{uv} and y_{us} are will be difficult to distinguish in this approach. This difference makes KG completion models unable to fully adapt to edge representation learning. [6] introduces a framework for learning a node embedding, a projection matrix, and an edge function jointly, so an edge can be represented by the concatenation of two *projected* node embeddings. However, this approach is limited in that it ignores the characteristic of multiple views [1] that is often found in social networks. It also fails to consider textual interactions, which are reflective of the social signals (e.g. strength and affinity) between nodes.

1.1 Research Questions

To address the deficiencies discussed in the previous section, in this dissertation we aim to either answer or provide insights to the following questions:

1. A social network user usually has multiple *roles* with regard to their different relationships. The lack of consideration of multiple roles in previous network representation learning models limits their applicability for real-world social networks. If a multi-relational network representation learning model can be proposed, to what extent will the proposed model be more effective than the state-of-the-art network representation learning models for social networks?

2. Social network data often contain rich *textual communication* among users. However, these communications have not been incorporated in previous network representation learning models. If a joint embedding model can be proposed to combine language and social interactions, that is, to learn a semantic embedding for both users' texts and attributes into a shared latent space, to what extent will the proposed model be more

effective (i.e., more predictive on evaluation tasks) than the state-of-the-art network representation learning models for social networks?

3. Social networks often contain multiple *edge types*, which yields multi-view contexts that need to be considered in the edge representation. Previous network representation learning, KG learning, and previous single-view edge representation models are not suited to exploit this characteristic of social networks. To fill this gap, we aim to provide a better definition of edge representation with respect to the multi-view nature of social networks. Moreover, if a multi-view edge representation learning model can be proposed, to what extent will the proposed model be more effective in predicting edge existences for multiple views and distinguishing edges in asymmetric views?

We conjecture that our proposed works will provide a better definition for relational representation learning (including node, edge, relation, and semantic embeddings) that incorporates textual communication in social networks, and mitigate the three deficiencies discussed in the previous section.

1.2 Main Hypothesis and Proposed Research

The goal of the research in this dissertation is to verify the following three hypotheses:

1. The proposed multi-relational network representation learning model, which incorporates textual interactions, produces learned representations that are *more predictive on evaluation tasks in social networks* compared to representations learned with corresponding state-of-the-art baselines.

2. The proposed joint semantic embedding model, which combines language and social interactions into a shared latent space, produces learned representations that are *more predictive on evaluation tasks in social networks* compared to representations learned with corresponding state-of-the-art baselines.

3. The proposed multi-view edge representation learning model, which incorporates textual communications, asymmetry, and edge types, produces learned representations

that are *more predictive on evaluation tasks in social networks* compared to representations learned with corresponding state-of-the-art baselines.

In this dissertation, we propose a comprehensive solution to address the three deficiencies respectively, divided into three components:

1. The first part is a relationship embedding learning approach for social networks.
2. The second part is a joint semantic embedding model combining language and social interactions into a shared latent space.
3. The third part is a multi-view edge representation learning for social networks.

Table 1.1. A summary of proposed network representation learning methods

Proposed Work	**Data**					**Learned Representation**			
	Network Structure	Text	Attribute	Single View	Multiple View	Node	Edge	Text	Relation
Chapter 3	■	■		■		✓			
Chapter 4	■	■	■		■	✓			✓
Chapter 5	■	■	■	■		✓		✓	
Chapter 6	■	■	■		■	✓	✓		

Note: The shaded cells indicate the type of data used in each component and checkmarks indicate the type of learned representations.

Our proposed works covers multiple dimensions of characteristics of social networks as summarized in Table 1.1. The "Data" column shows that we consider heterogeneous information from data sources and we also learn different types of network representations, as shown in the "Learned Representation" column.

The rest of this document is organized as follows. We introduce the background of important concepts about representation learning methodologies in Chapter 2. For proof-of-concept, we present an exploratory approach to incorporating textual communication into node representation learning in Chapter 3, which serves as motivation for our remaining work. We then discuss the proposed relationship embedding learning approach for social

networks in Chapter 4. In Chapter 5, we describe and evaluate our proposed joint semantic embedding model for users' texts and attributes in a shared latent space. In Chapter 6, we propose a multi-view edge representation learning for social networks. Finally, the conclusion and contributions of the various components are highlighted in Chapter 7.

2. BACKGROUND

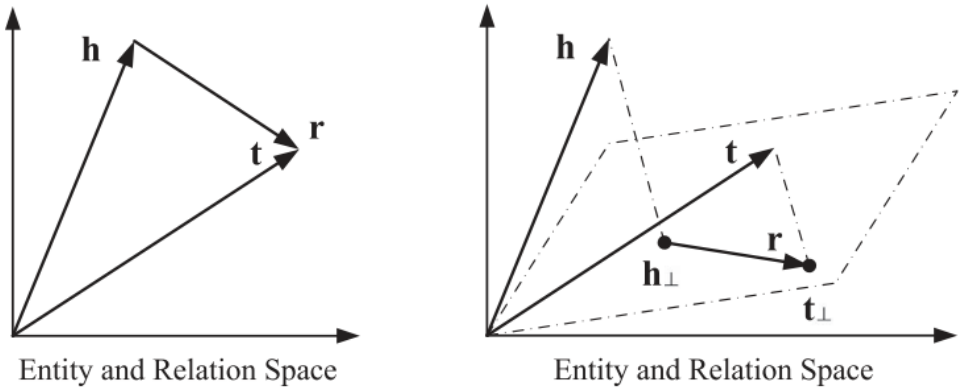
2.1 Knowledge-based Graph Completion

A Knowledge Graph (KG) is a multi-relational graph that consists of entities and relations. A triplet, which describes the connection between two entities with one relation, is defined as the format: (head entity, relation, tail entity). In order to represent the triplet of KG in low-dimensional vectors, many knowledge graph embedding approaches has been proposed and quickly gained massive attention [7]–[9], [22]–[25]. The key idea is to embed components of a KG including entities and relations into continuous vector spaces, so as to simplify the manipulation while preserving the inherent structure of the KG [11]. Those learned entity and relation embeddings can be treated as features in many tasks, such as KG completion [7], [8], relation extraction [26], [27], entity classification [22], [28], and entity resolution [22], [24].

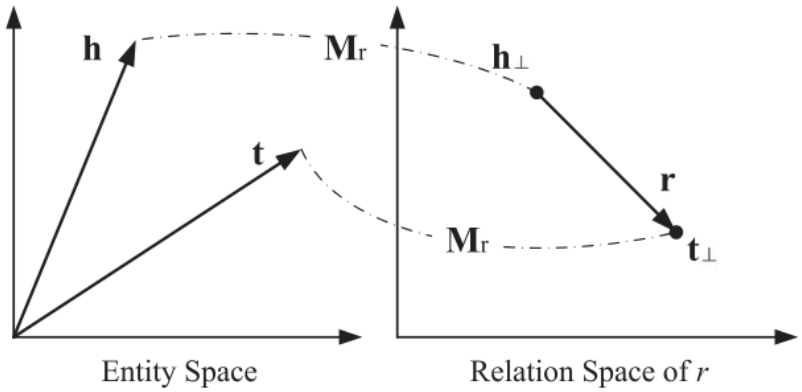
The previous studies adopted optimization techniques to maximize the plausibility of entity and relation embedding based on the observed triplets. Since the learning process only considered the triplets, the learned entity and relation embeddings might not be predictive for downstream tasks [29], [30]. Thus, many subsequent studies have focused on jointly leveraging other types of information, such as entity types [12], [31], relation paths [32]–[34], textual descriptions [8], [12], [35], [36], and even logical rules [29], [37], [38], in the optimization process to learn more predictive embeddings. According to [11], those KG models can be divided into two groups:

- *Translational Distance Models*, such as TransE[7], TransH[8], and TransR[9], which exploit distance-based scoring functions. As [11] illustrated in figures 2.1a, 2.1b and 2.1c, they measure the plausibility of a fact as the distance between the two entities, usually after a translation carried out by the relation.

- *Semantic Matching Models*, such as RESCAL [22], which exploits similarity-based scoring functions. They measure plausibility of facts by matching latent semantics of entities and relations embodied in their vector space representations.



(a) TransE. (b) TransH.



(c) TransR.

Figure 2.1. Simple illustrations of KG-based translation models.

2.2 Network Representation Learning

A network can be represented as a graph. For the rest of the chapter, we will use *network* and *graph* interchangeably. Traditionally, graph embeddings have been described in the context of dimensionality reduction [39]. Classical techniques for dimensionality reduction include principal component analysis (PCA) [40] and multidimensional scaling (MDS) [41]. Both methods seek to represent an $n \times m$ matrix M as a $n \times k$ matrix where $k \ll n$. For graphs, M is typically an $n \times n$ matrix, where could be the adjacency matrix, normalized Laplacian matrix or all-pairs shortest path matrix, to name a few. Both methods are capable of capturing linear structural information, but fails to discover the non-linearity within the input data. To fill the gap, there were two types of approaches proposed in literature: 1) Node representation learning and 2) edge representation learning. As shown in Figure 2.2 from

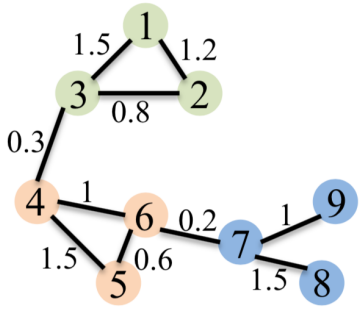
[42], there is a toy example of embedding a graph into a 2D space in different granularities, i.e., according to different needs, we may represent a node/edge as a low-dimensional vector. More details will be discussed in the following sections.

2.2.1 Node Representation

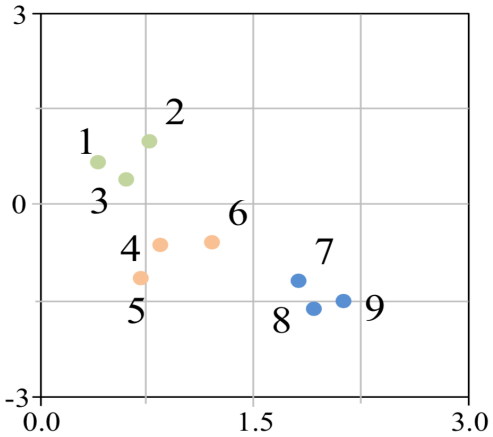
Node embedding is to encode nodes as low-dimensional vectors that summarize their graph position and the structure of their local graph neighborhood [43]. These low-dimensional embeddings can be viewed as encoding, or projecting, nodes into a latent space, where geometric relations in this latent space correspond to interactions (e.g., edges) in the original graph [44]. Many approaches have been proposed to learn node representation for data visualization, node classification, link prediction, and recommendation. DeepWalk [2] predicts the local neighborhood of nodes embeddings to learn graph embedding. LINE [3] learns feature representations in first-order proximity and second-order proximity respectively. GraRep [4] learns graph representation by optimizing k-step loss functions. Node2Vec [5] extends DeepWalk with a more sophisticated random walk procedure and explores diverse neighborhoods. SDNE [45] proposes a semi-supervised deep neural network model to captures the highly non-linear network structure. They jointly exploit the first-order proximity and second-order proximity to characterize the local and global network structure. [19] proposes a semi-supervised framework which simultaneously preserves network structure and relations between nodes by learning network embeddings with edge labels.

2.2.2 Edge Representation

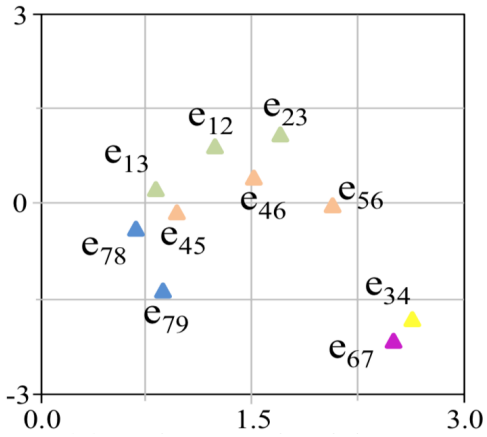
Graph edges are required to accurately describe relations between two nodes and are applicable to real world tasks, such as link prediction and link classification. Previous node embeddings works like Node2Vec [5] considers using binary operators, such as average, Hardmard product, L1 distance, and L2 distance to represent an edge; and [19] takes simple concatenation for constructing edge embeddings. However, these simple representations failed to describe the direction of edges [39]. For example, it represents two directed edges (u, v) and (v, u) with the same representation. Many works proposed the representations specific



(a) Input Graph G.



(b) Node Embedding.



(c) Edge Embedding.

Figure 2.2. A toy example of embedding a graph into 2D space with node or edge perspective.

to edges. WLNM [46] extracts an enclosing subgraph for each link and encodes the subgraph to an adjacency matrix as link's surrounding environment, namely the link's representation. [47] proposes an edge representation learning by K-means [48] to divide edges into communities by learning edge feature representations for identifying overlapping communities. [49] introduces edge role discovery and proposed a framework for learning and extracting edge roles from large graphs. [6] explicitly models edges as graph likelihood optimization of node representations in order to preserve directed edge information. [50] predicts edge types in multi-layer graphs that have various types of edges with regard to a node similarity which considered homophily and heterophily. They formulate the problem as simultaneously finding k special vectors (directions) and assigning one of them to every edge in order to maximize a scalar goodness function. Edge2Vec [51] trains an edge-type transition matrix by

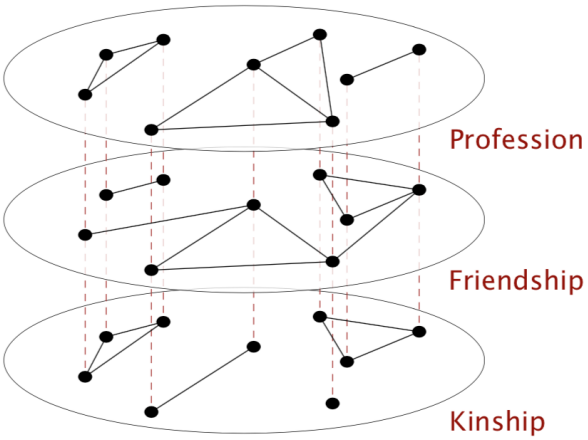


Figure 2.3. A toy multi-view network.

an Expectation-Maximization approach, and learned node embeddings on a heterogeneous graph via the trained transition matrix with a stochastic gradient descent model. ENRNM [52] combines node and edge representations via learning from the formation mechanism of graph links. They train a fully connected neural network with the learned node and edge representations for link prediction.

2.3 Multi-View Network Representation Learning

Although network representation learning is empirically efficient for many networks, most of these approaches assume there only exists a single type of proximity between nodes in a network. However, in reality, a network usually has multiple types of proximities or relations between nodes. For example, Figure 2.3 from [53] shows a toy example of multi-view network where each node represents a person and the three views represent three types of interpersonal relations. Thus, many previous studies [1], [53]–[67] focus on learning node representations for networks with multiple views for adapting their models to the real world datasets.

Some works consider multiple signals in networks. [54] learns user representations using deep learning by considering rich linguistic and network evidence gathered from social media. The algorithm combines noisy heterogeneous cues, such as the text a person writes, his/her

attributes and social relations to other people. GCCA [55] learns multi-view user embeddings for Twitter users with respect to many aspects of a user's online activities and posts.

Matrix factorization is adopted in several models. [56] proposes a collective matrix factorization model that simultaneously factor several matrices, sharing parameters among factors when an entity participates in multi-type relations. [57] presents an unsupervised algorithm for combining information from related views, using a late integration strategy. Combination is performed by applying a matrix factorization approach to group related clusters produced on individual views. [58] proposes a multi-view clustering algorithm based on non-negative matrix factorization (NMF) by searching for a factorization that gives compatible clustering solutions across multiple views. They develop a joint matrix factorization algorithm to incorporate individual matrix factorization and inconsistency between each view's coefficient matrix and the consensus. [59] introduces an unsupervised method for integrating multiple relation-based or feature-based views to produce a single unified graph representation, based on the combination of the k-nearest neighbour sets for users derived from each view.

Several studies design clustering approaches to tackle this problem. [60] develops a multi-view spectral clustering via generalizing the normalized cut from a single view to multiple views. They build multi-view transductive inference on the basis of multi-view spectral clustering and lead to a mixture of Markov chains defined on every graph. [61] assumes the views are (conditionally) uncorrelated, conditioned on which mixture component generated the views. Under the assumption, they develop a subspace learning method, based on Canonical Correlation Analysis, to project the data into a lower dimensional subspace. [62] proposes a spectral-embedding algorithm, multi-view spectral embedding, which learns a low-dimensional and smooth embedding over all views simultaneously. In order to exploit information from a clustering of multiple views rather than using individual views, [63] presents a spectral multi-view clustering framework by philosophy of co-regularization. The co-regularized spectral clustering has a joint optimization function for spectral embeddings of all the views. An alternating maximization framework reduces the problem to the standard spectral clustering objective which is efficiently solvable using state-of-the-art eigensolvers. [64] considers vector-based representation cannot fully represent the multi-view graphs, so they propose a multi-view clustering framework on graph instances with Graph Embedding

(MCGE). Specifically, They model the multi-view graph data as tensors and apply tensor factorization to learn the multi-view graph embeddings.

A few studies jointly learn embeddings via optimization. MVN2VEC [53] learns embeddings for multi-view networks by simultaneously modeling two characteristics of network: preservation and collaboration. They perform an optimization with two models MVN2VEC-CON and MVN2VEC-REG to joint consider the two characteristics. They adopt the random walk and the skip-gram approach to formulate the intra-view loss function. [1] learns node representations for networks with multiple views, which aims to infer robust node representations across different views. They propose a multi-view representation learning approach, which promotes the collaboration of different views and lets them vote for the robust representations by an attention mechanism, which enables each node to focus on the most informative views. [67] proposes a network embedding method, intra-view and inter-view attention for Multi-view Network Embedding (I2MNE), which leverages both the multi-view network structure and the node features to generate node representations. They introduce the intra-view attention when aggregating node features from neighbors for each single view and the inter-view attention when integrating representations across different views.

Recently, deep neural network (DNN) has gained increasing popularity on the field of multi-view representation learning. [66] proposes a co-regularized Deep Multi-Network Embedding (DMNE) method, which manipulates cross-network relationships to reinforce the learning of node embeddings in different networks. [65] considers learning representations in the setting in which they have access to multiple unlabeled views of the data for representation learning while only one view is available at test time. They compare several approaches, such as Split Auto-Encoder (SplitAE) [68], Correlated Auto-Encoders (CorrAE) [65], Deep Canonical Correlation Analysis (DCCA) [69], Deep Canonically Correlated Auto-Encoders (DCCAE) [65], and minimum-Distance Auto-Encoders (DistAE) [65]. Their results suggest an advantage for correlation-based representation learning, while the best results on most tasks are obtained with DCCAE.

For edge representation, [20] proposes a friend recommendation system (an application of link prediction) using edge embedding in social networks. They mined network representa-

tion that exploits edge heterogeneity in multi-graphs. However, this work does not consider edge direction and obtain edge vectors from concatenation of node embeddings directly.

3. COMBINING LANGUAGE AND SOCIAL INTERACTIONS INTO A SHARED REPRESENTATION

3.1 Introduction

The interactions, social bonds and relationships between people have been studied extensively in recent years. Broadly speaking, these works fall into two, almost completely disconnected, camps. The first, focusing on *social network analysis*, looks at the network structure and information flow on it as means of inferring knowledge about the network. For example, works by [70], [71] model the evolution of network structure over time, and works such as [72], [73] use the network structure to predict properties of links (e.g., strength, sign).

The second camp, focusing on *natural language analysis*, looks into tasks such as extracting social relationships from narrative text [14]–[16] and analyzing the contents of the information flowing through the network. For example, works by [17], [74]–[79] extract attributes of, and social relationships between, nodes by analyzing the textual communication between them. Other works [80], [81] use the social network to inform language analysis.

Both perspectives on social network analysis resulted in a wide range of successful applications; however, they neglect to model the interactions between the social and linguistic representations and how they complement one another. One of the few exceptions was discussed in [17], which inferred sentiment links between nodes in a social network by jointly modeling the local output probabilities of a sentiment analyzer looking at the textual interactions between the nodes and the global network structure. While resulting in better performance, inference is done over two *independent* representations, one capturing the linguistic information, and the other, the network structure.

Instead, in this work we take the first step towards finding a joint representation over both linguistic and network information, rather than treating the two independently. We follow the intuition that interactions in a social network can be fully captured only by taking into account both types of information together. To achieve this goal, we embed the input social graph into a dense, continuous, low-dimensional vector space, capturing both network and linguistic similarities between nodes. Word [82], [83] and network [2], [3] embedding approaches that were recently proposed, aim to combat a similar problem

in their respective domains–data sparsity. Both follow a similar approach–embed discrete objects (words or nodes in the graph) into a continuous vector representation, based on the context they appear in. Our approach aims to map both social and linguistic information into the same vector space, rather than embedding the two aspects into two independent spaces. The social graph, originally containing only quantitative properties of the interaction between nodes (e.g., number of messages exchanged between nodes), is extended to capture the contents of these interactions, by computing the textual similarity between the messages generated by each one of the nodes. The computed similarity is used to weight the edges between adjacent nodes. We embed the modified graph nodes into a vector space, using the embedding technique described by [3].

We evaluate the joint representation by using it in two social relationship prediction tasks and comparing it to several different word-based and network based representations. Our experiments show the advantage of the joint representation.

3.2 Problem Formulation

Our primary assumption is there is a latent space that influences the interactions we observe among people. Thus the goal of our work is to learn this latent representation from the observed data. We describe the data and problem more specifically below.

3.2.1 Data

We assume that the data comprise a graph $G = (V, E)$, where nodes V correspond to entities (e.g., users in a social network), and the edges E correspond to textual interactions among the entities (e.g., emails, messages). Each edge $\mathrm{e}_{\mathrm{ij}}^{t} \in E$, which refers to a message sent from node v_{i} to node v_{j} at time t, has an associated document representation d_{ij}^{t}. We refer to the set of messages (documents) between nodes v_{i} and v_{j} as $\mathbf{E}_{\mathrm{ij}} := \{\mathrm{e}_{\mathrm{ij}}^{t}\}_{t}$ ($\mathbf{D}_{\mathrm{ij}}$ respectively). Moreover, we refer to the set of messages (documents) sent by a node v_{i} to any other node as $\mathbf{E}_{\mathrm{i}} := \{\mathrm{e}_{\mathrm{ij}}^{t}\}_{t,\mathrm{j}}$ ($\mathbf{D}_{\mathrm{i}}$ respectively).

3.2.2 Motivation

Given this type of network data, the goal is to discover the underlying latent representation of the nodes. Our assumption is that the entities are embedded in a latent space that influences the frequency and nature of their communication. We assume that each node has a location in space (e.g., in $\mathbf{R}^2$, the location of v_i is $\mathbf{v}_\mathrm{i} := (x_\mathrm{i}, y_\mathrm{i})$), and that pairwise node distances (e.g., $d(\mathbf{v}_\mathrm{i}, \mathbf{v}_\mathrm{j})$) affect the likelihood of communication and the content of that communication. More specifically, we assume that nearby nodes are more likely to communicate, and talk about similar things. Thus, we assume the latent space embedding represents entities' interests and pairs of entities with similar interests are more likely to interact. These assumptions are motivated by online communities where users exhibit *homophily* [84], i.e., users with common interests are more likely to form relationships.

3.2.3 Problem Definition

Given the framework and assumptions described above, we can now state the problem definition for this work. Assume as input, a multi-graph $G = (V, E)$ with messages between nodes in the graph that can be modeled as a set of documents. The goal is to learn an embedding of the nodes V in $\mathbf{R}^\mathbf{k}$ such that the representation reflects both the frequency and content of the messages.

To achieve this we will consider several different ways to compute the embedding based on optimizing (1) network connectivity, (2) message content, and (3) connectivity and content. Our conjecture is that jointly considering connectivity and content will produce an embedding that is more robust to noisy interaction data. Strong (but introverted) friends may talk less frequently but share more common interests, compared to gregarious users who talk more frequently but with many (weak) friends.

Since there is no ground truth for quantitative evaluation, it is difficult to directly evaluate the quality of a learned embedding. Thus, we evaluate our methods indirectly via related classification tasks. In this work, we will use the learned embeddings in two link-based prediction tasks, where we differentiate (1) strong vs. weak(er) friendships, and (2) employees working in the same vs. different groups.

3.3 Method

The input for our task is the text-enriched network graph G. The goal is to compute a node embedding from G and then use the embedding to generate features for pairs of nodes, which can then be used for a prediction task. The process follows these steps.

1. **Textual-Similarity (TS) Infused Social Graph:** Construct graph weights $\mathbf{W}_{ij}$ based on the text in G, according to **(1)** a *Node* or *Edge* view of the documents, and **(2)** using *Topic Model* or *Word Embedding* to represent the content.

2. **Node Embedding:** Construct an embedding function $V \rightarrow R^k$, mapping the (weighted) graph nodes into a R^k dimensional space. We used the LINE method [3]. We omit the details due to space restrictions.

3. **Feature Extraction:** Construct a feature set for each node pair, using 9 similarity measures between the nodes' k-dimensional vector representations from the embedding. We experiment with additional features extracted directly.

3.3.1 Creating the TS-Infused Social Graph

The TS-Infused social graph captures the interaction between node pairs by modifying the strength of the edge connecting them according to the similarity of the text generated by each one of the nodes. We identify several design decisions for the process.

Node vs. Edge

Each edge $e_{ij} \in G$ is associated with textual content d_{ij}. We can characterize the textual content from the point of view of the *node* by aggregating the text over all its outgoing edges (i.e., $\mathbf{D}_i$), or alternatively, we can characterize the textual content from the edge point of view, by only looking at the text contained in the relevant outgoing *edges* (i.e., $\mathbf{D}_{ij}$).

Representing Textual Content using Topic Models vs. Word Embedding

Before we compute the similarity between the content of two parties, we need a vector space model to represent the textual information (the set of documents $\mathbf{D}_{\mathrm{i}}$, or $\mathbf{D}_{\mathrm{ij}}$). One obvious method for this is topic modeling, in which the textual content is represented as a topic distribution. In this approach, we learn a topic model over the set of documents, and then represent each document via a set of topic weights ($\mathbf{T}_{\mathrm{i}}$ or $\mathbf{T}_{\mathrm{ij}}$). An alternative approach is using word embedding, which has been proved effective as a word representation. In this approach, we represent each document as the average of the embedding over the words in the document ($\mathbf{WE}_{\mathrm{i}}$ or $\mathbf{WE}_{\mathrm{ij}}$). Given the distributional representation of text associated with a node/edge, we assign a weight (w_{ij}) for each edge (e_{ij}) as the cosine similarity between vector representation of contents from neighboring nodes (e.g., $d(\mathbf{T}_{\mathrm{i}}, \mathbf{T}_{\mathrm{j}})$ or $d(\mathbf{T}_{\mathrm{ij}}, \mathbf{T}_{\mathrm{ji}})$, where d is cosine similarity).

3.3.2 Node Embedding

We utilize the LINE embedding technique [3], aimed at preserving network structures when generating node embedding for social and information networks. LINE uses edge weights corresponding to the number of interactions between each pair of nodes. This only makes use of the network structure, without taking advantage of the text in the network. We modify the embedding procedure by using the edges weights $\mathbf{W}_{\mathrm{ij}}$ described above (i.e., based on the cosine similarity of the text between nodes i, j) and use the LINE algorithm to compute a k-dimensional embedding of the nodes in G.

3.3.3 Feature Extraction

Distance-based Features

Given a node pair represented by their k-dimensional node embedding, we generate features for the pair according to nine similarity measures. The nine measures used by us are Bray-Curtis distance, Canberra distance, Chebyshev distance, City Block (Manhattan)

distance, Correlation distance, Cosine distance, Minkowski distance, Euclidean and squared Euclidean distance.

Additional Features

Besides the distance-based features, we can also add one or more other basic features related to nodes in the network. These include the following: (1) Network: The number of interactions between two nodes, e.g. number of emails sent and received. (2) Unigram: The unigram feature vector for text sent for each node. (3) Word embedding features: The word embedding vector for text sent for each node. Again we use the average of word embedding to represent documents.

3.4 Experimental Results

3.4.1 Datasets

Purdue Facebook Network

We analyzed the public Purdue Facebook network data from March 2007 to March 2008, which includes 3 million post activities. Members can set friends as top (close) friends to get the timely notifications without a confirmation by the other. We collected 945 mutually top friend pairs for two users who set each other as top friend and 34633 one-way top friend pairs if there is only one of them set the other as top friend. The dataset will be referred as "Facebook" in this work. We evaluated our method by a classification task of the two different social relationships.

Avocado Email Collection

This collection consists of 279 e-mail accounts, from which we extracted the job titles and departments of 136 accounts. We divided these accounts into three groups, according to their positions in the company, namely executives, engineering department, and business department. We will refer to this dataset as "Avocado" in this work. The task is defined as predicting whether two accounts belong to the same group. In order to make use of text

signal. We will only consider account pairs that have correspondence between each other. There are 2232 positive and 1409 negative examples in this dataset.

Table 3.1. Prediction results over the two datasets (F1 score)

Dataset	**Embedding**	$\emptyset$	**N**	**W**	**WE**	**N+W**	**N+WE**	**N+W+WE**
Facebook (F1)	no GE	49.45	77.80	75.04	75.09	81.23	79.14	79.26
	GE	53.36	78.54	75.82	75.68	82.09	79.11	78.39
	GE^N_{TM}	61.58	80.16	76.33	76.31	82.69	78.72	79.68
	GE^E_{TM}	78.36	80.51	77.51	77.51	80.23	79.82	80.38
	GE^N_{WE}	59.81	79.98	75.44	76.82	81.19	79.62	79.15
	GE^E_{WE}	80.66	82.49	81.96	81.07	83.31	82.06	**83.72**
Avocado (F1)	no GE	49.69	40.91	55.03	57.89	53.33	56.64	55.36
	GE	65.75	66.15	65.77	66.57	66.24	66.99	66.53
	GE^N_{TM}	66.66	66.65	66.49	**67.28**	66.83	67.12	66.84
	GE^E_{TM}	63.09	64.67	64.79	64.09	64.80	65.05	64.49
	GE^N_{WE}	61.33	64.51	64.60	63.83	65.46	64.67	65.39
	GE^E_{WE}	52.03	55.11	56.94	56.03	57.40	57.18	58.48

3.4.2 Classification Results

Using the features defined in the previous section, we train Logistic Regression classifier via scikit-learn in Python. We show the ten-fold cross-validation performance of our features on Facebook and Avocado datasets in Table 3.1. It represents the results of five different approaches to generate node embedding, and with or without adding additional features. GE is the original embedding method, the superscript N or E represent the Node or Edge, and the subscript TM or WE represent Topic Model or Word Embedding used to construct the TS-Infused graph respectively. The N, W, WE in the columns indicate the Network, Unigram and Word embedding as additional features. $\emptyset$ with no GE shows the result of random generated embedding. In this work, we use LINE as the node embedding method, Latent Dirichlet Allocation [85] for topic modeling with ten topics and Skip-Gram for word embedding. The regularization parameters are optimized. Since the Facebook and Avocado

datasets are unbalanced, we randomly downsamples the majority class to equate the size of both classes. The results show in the Table are the average scores of ten different random downsampling.

For Facebook dataset, the results of all embeddings constructed by TS-Infused social graph outperforms the original embedding GE. It shows the joint representation over linguistic information and network structure is more effective than only considering one of them independently. The results on Avocado dataset also confirm the advantage of shared representation. GE^N_{TM} significantly outperforms other text-based or network-based methods. The performance of aggregating text sent by a node is better than only looking at text on one outgoing edge, which is opposite to the results on Facebook dataset. This could be resulted from the difference between two prediction tasks. In the Facebook dataset, we try to distinguish strong and weak(er) friendship, in which case the messages they sent to each other are most indicative. While when we predict whether two persons belong to the same group inside a company, the interaction they had with their colleagues would tell us more about the community they are from.

3.5 Discussion

Despite the clear inter-dependency between analyzing the interactions in social networks, and analyzing the natural language content of these interactions, these aspects are typically studied independently. The main contribution in this preliminary work is to present a first step towards finding a joint representation, by embedding the two aspects into a single vector space. We show that the new representation can help improve performance in two social relations prediction tasks.

4. RELATIONSHIP EMBEDDING IN SOCIAL NETWORKS

4.1 Introduction

Representation learning has been applied widely in different areas to extract useful information from data when building classifiers for inferring node attributes or predicting links in graphs. Many previous studies proposed low-dimensional network embeddings to learn graph representations [2]–[5], [45]. When applied to social networks, these models project users to a hyperspace to capture the relational and structural information conveyed by the graph. However in social networks, because a user often has different roles for different relationships, learning a single unique representation for all users/relations may not be effective. For example, a user could be close to a one set of friends because they were college classmates but close to another because they are colleagues at work. To capture this information, it is important to consider the characteristics of relationships between users when learning representations of social networks.

Knowledge graphs are multi-relational graphs that are composed of entities as nodes and relations as different types of edges. An edge instance is a triplet of fact (head entity, relation, tail entity). There has been a surge of interest in learning graph representations of social networks by simultaneously learning user and relationship embeddings based on the concept of triplet [7]–[10]. These methods have considered both network structure and node relations to improve the quality of embedding. At the same time, semantic content of entities can also provide abundant information for representation learning. [12], [13] take the description of entities into account to incorporate text into embedding learning. However, typically users' descriptions do not provide much information about the relationships between pairs of users.

In this work, we make the observation that social network data often contain *textual communications* among users, and that this information is a valuable signal about the types and strength of relationships between users. However, to date this information has not been used effectively in network embedding methods. To address this, we propose a novel relationship embedding model, *TransConv*.

TransConv is a structural embedding approach using relation hyperplanes, where every relationship can be viewed as a translation of users in the embedding space. To incorpo-

rate textual communication into the learned embeddings, we develop two different types of *conversation factors* to include in the objective function when learning the embeddings. Our work was inspired by Word2Vec word embedding model [86] and knowledge graph completion models [7], [8]. Word2Vec allows people to use vector arithmetic to work with word analogies, for instance, $King - Man + Woman \approx Queen$. This can be interpreted to mean that the relationship between *King* and *Queen* is similar to the one between *Man* and *Woman*. Instead of working with analogies, our model will directly learn vector representations of relationships between users in social networks. Therefore, we aim to leverage ideas from the knowledge graph completion problem to jointly learn representations of entities and relations. We extend previous approaches by incorporating conversation-based factors to improve the learning process. We evaluate *TransConv* on three different classification tasks: social network completion, triplets classification, and multilabel classification. The experimental results show that our approach outperforms other state-of-the-art models on two real-world social network datasets, and notably it improves prediction accuracy for both frequent and infrequency relations.

4.2 Problem Formulation

In social network data, we have a set of users ($U = \{u_{\mathrm{i}}\}$), user attributes ($\mathbf{X} = \{X_{\mathrm{i}}\}$) collected from user profiles and their group memberships, and messages exchanged among the users ($D = \{d_{\mathrm{ij}}\}$). More specifically document $d_{\mathrm{ij}} \in D$ represents the set of posts t_{ij} sent from u_{i} to u_{j}. Relationships between pairs of users are either defined by the attribute values that they share in common (e.g., $r_k(\mathrm{i},\mathrm{j}) = 1$ if $x_{\mathrm{i}} = x_{\mathrm{j}} = k$ and 0 otherwise), or defined by certain directional attributes (e.g., u_{j} is u_{i}'s top friend, u_{j} is senior to u_{i}). Given attribute values of interest in the data, we define a set of relations ($R = \{r_k\}$).

Given a pair of user and their relation as a triplet (u_{i}, r, u_{j}), the goal of this work is to learn a joint embedding for users and relationships, such that every relation can be viewed as a translation of users in the embedding space. Let $\hat{u}_{\mathrm{i}}$ and $\hat{u}_{\mathrm{j}}$ denote the user embeddings of u_{i} and u_{j}, and $\hat{r}$ denote the relationship embedding of r. The embedding $\hat{u}_{\mathrm{i}}$ is close to $\hat{u}_{\mathrm{j}}$ by adding the relationship embedding $\hat{r}$ (i.e., $\hat{u}_{\mathrm{i}} + \hat{r} \approx \hat{u}_{\mathrm{j}}$). The embeddings of users and relationships are in the same space $\in \mathbb{R}^k$. Let Δ denote the set of golden (positive; true)

triplets, for which the relationship holds in the data and $\Delta_{(u_i,r,u_j)}$ stand for the set of negative triplets constructed by corrupting a golden triplet (u_i, r, u_j).

Previous work on structural embedding began with TransE [7], which first adopts the concept to learn entity embedding of knowledge bases (KB). TransE [7] assumes the error $||\hat{u}_1 + \hat{r} - \hat{u}_2||_{l1/l2}$ is low if (u_1, r, u_2) is a golden triplet. This works well for irreflexive and 1-to-1 relations but fails to deal well with reflexive, N-to-1, 1-to-N or N-to-N relations. TransH [8] addresses the issues of TransE by introducing relation-specific hyperplanes w_r. Several models, such as TransR [9] and TransD [10], then extend TransH and enhance the embedding performance by learning mapping matrices to relation spaces.

However, the previous work only considers the network structure among entities and ignore the textual information content in messages. In this work, we aim to exploit the message information among users to improve the learned embedding, e.g., by automatically identifying content relevant to particular relations. Specifically, when modeling people's relationships in social networks, we consider a sophisticated model to utilize the "interaction" between two users rather than design a complicated hyperplane projection. For example, u_1, u_2, and u_3 are three users who described themselves as supporters for the same political party, but (u_1, u_2) discuss politics extensively and (u_1, u_3) rarely discuss it. Let's denote $r_{politics}$ as "the same political party". If we model the relation $r_{politics}$ by TransH or its extended models, they treat the triplets $(u_1, r_{politics}, u_2)$ and $(u_1, r_{politics}, u_3)$ in the same way (because they do not consider the content of discussion between users). In contrast, our approach will focus more on (u_1, u_2) than (u_1, u_3) when learning the embedding, under the assumption that the frequent discussion indicates a stronger relationship with respect to $r_{politics}$.

More specifically, to incorporate interaction information in the embedding, we define two new conversational factors to use during learning: *conversation similarity* and *conversation frequency* (defined below). Using these new factors, we then outline our novel relationship embedding model, *TransConv*.

4.2.1 Conversation Similarity Factor

To capture the textual similarity of the interaction between a user pair regarding a particular relation, we define a *conversation similarity* factor μ_{ij}^r. The factor represents the

textual similarity of the interaction between a user pair $(u_{\mathrm{i}}, u_{\mathrm{j}})$ with respect to relation r, based on the documents d_{ij} and d_{ji} (the collection of messages between u_{i} and u_{j}). We compute μ^r_{ij} as follows:

1. First, we identify the most representative set of words for each relation $r \in R$. To do this, we collect the set of pairs $(u_{\mathrm{i}}, u_{\mathrm{j}})$ with relation r and concatenate all their posts into a single (large) document D_r. We repeat this process for each of the relations in R. From the resulting documents, we compute the TF-IDF [87] values for each word in each document D_r. TF-IDF scores are widely used as a numerical statistic to reflect how important a word is to a document in a collection. Then for each document D_r, we identify the top-K words with largest TF-IDF values and use those as the representative words as the dictionary W_r for the relation r.

2. Next, we compute a word existence vector (denoted as $wv_{r_{\mathrm{ij}}}$ and $wv_{r_{\mathrm{ji}}}$) based on the dictionary W_r to transform the textual interactions between u_{i} and u_{j} with regard to relation r. This tracks whether the pair has used the words that are representative to the relation r. For each word w in W_r, the value is set to 1 if w exists in the posts d_{ij} (or d_{ji}), otherwise it is set to 0.

3. Finally, we use the word existence vectors to compute the conversation similarity factor for each user pair using the similarity function SIM (e.g., cosine similarity): $\mu^r_{\mathrm{ij}} = SIM(wv_{r_{\mathrm{ij}}}, wv_{r_{\mathrm{ji}}})$. This tracks whether the pair uses similar words from the relation r in their communication back and forth. We repeat this for every $r \in R$.

The similarity factor μ^r_{ij} measures whether u_{i} and u_{j}'s mutual discussion is relevant to r, and evaluates the degree of affinity between u_{i} and u_{j}.

4.2.2 Conversation Frequency Factor

We define a *conversation frequency* factor ϕ^r_{ij} to represent the strength of the interaction between a user pair $(u_{\mathrm{i}}, u_{\mathrm{j}})$ with respect to relation r. In this factor, we also use the relation dictionaries $\mathbf{W_r}$ from steps 1-2 above.

We first define $out^r(u_\mathrm{i}, u_\mathrm{j})$ as the sum of fraction of words in dictionary W_r used in the messages from u_i to u_j:

$$out^r(u_\mathrm{i}, u_\mathrm{j}) = \sum_{p=1}^{m} \frac{|w_p^r|}{|w_p|} \tag{4.1}$$

Here m is the number of messages from u_i to u_j. w_p is the set of words used in message p. w_p^r is the intersection of w_p and W_r. Note that the more u_i communicates with u_j, using words relevant to relation r, then the value of $out^r(u_\mathrm{i}, u_\mathrm{j})$ will be larger.

Next, we define the conversation frequency factor ϕ_ij^r, which reflects the intensity of interaction between two users with respect to relation r, compared to other users:

$$\phi_\mathrm{ij}^r = \frac{out^r(u_\mathrm{i}, u_\mathrm{j})}{\sum_{k=1}^{n} out^r(u_\mathrm{i}, u_k)}, \forall u_k \in \{u_1, u_2, ..., u_n\} \tag{4.2}$$

If u_i interacts more frequently with u_j compared to other users, then the frequency factor will be larger. The factor can also distinguish whether the interaction between u_i and u_j is one-way or two-way.

After computing the above factors $\{\mu_\mathrm{ij}^r\}$ $\{\phi_\mathrm{ij}^r\}$ for each relation, we will use them to weights the errors of triplets used in the embedding objective. We do not consider the documents D further.

4.3 TransConv: Translating on Conversation

In our *TransConv* model, we assume that people who have similar (stronger) textual interactions would share similar (stronger) relationships, which can be used to improve the learned embeddings. That is, their relationships can be translated better with the aid of their conversations. To achieve this goal, we jointly incorporate the conversation similarity factors $\{\mu_\mathrm{ij}^r\}$ and frequency factors $\{\phi_\mathrm{ij}^r\}$ introduced in last section when learning user and relationship representations.

For a triplet (u_i, r, u_j), we learn the relationship-specific hyperplane w_r for relation r as well as the user embeddings $\hat{u}_\mathrm{i}$ and $\hat{u}_\mathrm{j}$ by projecting users on the relationship hyperplane.

The projections are denoted as $\hat{u}_{\mathrm{i}_\perp}$ and $\hat{u}_{\mathrm{j}_\perp}$, respectively. If ($u_{\mathrm{i}}$, r, u_{j}) is a golden triplet, the aim is to ensure that $\hat{u}_{\mathrm{i}_\perp}$ and $\hat{u}_{\mathrm{j}_\perp}$ are connected by a translation vector $\hat{r}$ on the hyperplane with low error measured by $||\hat{u}_{\mathrm{i}_\perp} + \hat{r} - \hat{u}_{\mathrm{j}_\perp}||_{l_{1/2}}$.

We define a score function $f_r(u_{\mathrm{i}}, u_{\mathrm{j}})$ to assess the quality of the embeddings for u_{i} and u_{j} w.r.t. relation r, and weight the score using their conversation similarity and frequency factors:

$$f_r(u_{\mathrm{i}}, u_{\mathrm{j}}) = [1 + \alpha\mu^r_{\mathrm{ij}} + (1-\alpha)\phi^r_{\mathrm{ij}}] \cdot ||\hat{u}_{\mathrm{i}_\perp} + \hat{r} - \hat{u}_{\mathrm{j}_\perp}||_{l_{1/2}} \tag{4.3}$$

Here α is a tunable parameter for assigning different learning weights to the similarity factor μ^r_{ij} and frequency factor ϕ^r_{ij}. The two factors play important roles augmentating the score function f_r. By constraining $||w_r||_2 = 1$, we formulate $\hat{u}_{\mathrm{i}_\perp}$ and $\hat{u}_{\mathrm{j}_\perp}$ as:

$$\begin{aligned} \hat{u}_{\mathrm{i}_\perp} &= \hat{u}_{\mathrm{i}} - w_r^T \hat{u}_{\mathrm{i}} w_r \\ \hat{u}_{\mathrm{j}_\perp} &= \hat{u}_{\mathrm{j}} - w_r^T \hat{u}_{\mathrm{j}} w_r \end{aligned} \tag{4.4}$$

Then the score function $f_r(u_{\mathrm{i}}, u_{\mathrm{j}})$ can then be rewritten as:

$$\begin{aligned} f_r(u_{\mathrm{i}}, u_{\mathrm{j}}) =& [1 + \alpha\mu^r_{\mathrm{ij}} + (1-\alpha)\phi^r_{\mathrm{ij}}] \cdot \\ & ||(\hat{u}_{\mathrm{i}} - w_r^T \hat{u}_{\mathrm{i}} w_r) + \hat{r} - (\hat{u}_{\mathrm{j}} - w_r^T \hat{u}_{\mathrm{j}} w_r)||_{l_{1/2}} \end{aligned} \tag{4.5}$$

The score is expected to be lower for golden triplets and higher for negative triplets. Since golden triplets with affinity (i.e., higher *similarity*) and stronger (i.e., higher *frequency*) interactions are weighted more heavily in the objective, the optimization will pay more attention to reducing the translation error for those triplets.

The concept of *TransConv* is illustrated in Figure 4.1. We simultaneously learn the user embeddings for u_1 and u_2 as well as the relationship embeddings for r_{senior_to} and $r_{christian}$. When u_1 and u_2 have more conversations related to a certain relation, *TransConv* minimizes the score $f_r(u_1, u_2)$ further. In other words, if u_1 and u_2 have two relations r_{senior_to} and $r_{christian}$, but they use more words relevant to $r_{christian}$ compared to r_{senior_to}, *TransConv*

will attempt to minimize $f_{r_{christian}}(u_i, u_j)$ more than $f_{r_{senior\_to}}(u_i, u_j)$. As illustrated in Figure 4.1, during the training phase, $f_{r_{christian}}(u_i, u_j)$ (i.e., the distance of the red double-headed arrow) is minimized compared to $f_{r_{senior\_to}}(u_i, u_j)$ (i.e., the distance of the blue double-headed arrow).

By considering projections on relational hyperplanes along with the augmentation of our proposed conversation factors, *TransConv* can encode different representations for each user, which depends on his/her relationships with others as well as the similarity and frequency of their textual discussions.

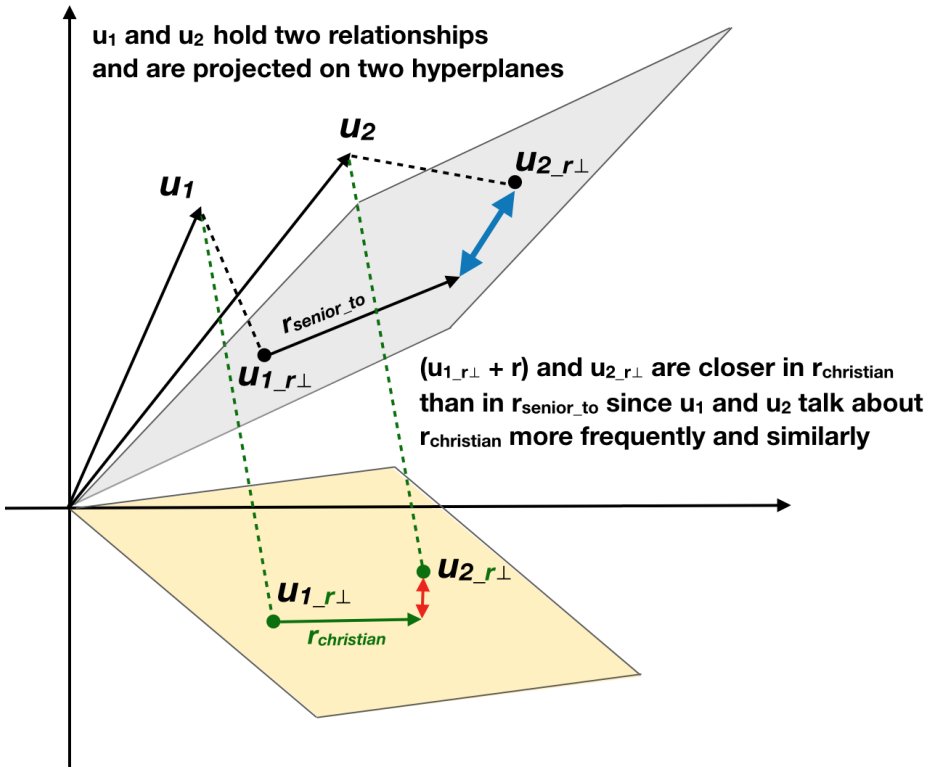


Figure 4.1. Simple illustration of *TransConv*.

4.3.1 Optimization

In order to maximize the difference between golden triplets and negative triplets, we define our loss function as:

$$L = \sum_{\substack{(u_{\mathrm{i}},r,u_{\mathrm{j}})\in\Delta \\ (u_{\mathrm{i}},r,u_{\mathrm{j}})\in\Delta_{(u_{\mathrm{i}},r,u_{\mathrm{j}})}}} [f_r(u_{\mathrm{i}},u_{\mathrm{j}}) + \gamma - f_r(u_{\mathrm{i}},u_{\mathrm{j}})]_+ \tag{4.6}$$

Here $[x]_+ \triangleq max(x, 0)$ and $\gamma > 0$ is the discriminative margin separating golden and negative triplets. The loss function sums over a corrupted negative triplet for each golden triplet (described more below). We adopt stochastic gradient descent (SGD) to minimize the above loss function. When minimizing the loss function, we enforce constraints as $\forall u \in U$, $||u||_2 \leq 1$ and $\forall r \in R$, $||w_r||_2 = 1$.

Initially, we construct the sample data from only golden triplets in Δ. In order to reduce false negative instances, we follow TransH [8] and apply Bernoulli sampling method to sample negative triplets. For each golden triplet (u_{i}, r, u_{j}) in Δ, our approach samples one negative triplet from $\{(u_{\mathrm{i}}, r, u_{\mathrm{j}}) \mid u_{\mathrm{i}} \neq u_{\mathrm{i}}, u_{\mathrm{i}} \in U\} \cup \{(u_{\mathrm{i}}, r, u_{\mathrm{j}}) \mid u_{\mathrm{j}} \neq u_{\mathrm{j}}, u_{\mathrm{j}} \in U\}$ and adds it to $\Delta_{u_{\mathrm{i}},r,u_{\mathrm{j}}}$. We assign different probabilities for replacing the head user (u_{i}) or the tail user (u_{j}) when corrupting the triplet, which depend on the mapping property (i.e., 1-to-N, N-to-1, and N-to-N) of the relation. Among all the triplets of a relation r, let *tph* denote the average number of tail users per head user and *hpt* denote the average number of head users per tail user. Then we define a Bernoulli distribution with parameter $\frac{tph}{tph+hpt}$ for sampling: given a golden triplet $(u_{\mathrm{i}}, r, u_{\mathrm{j}})$, we corrupt the triplet by replacing the head user with probability $\frac{tph}{tph+hpt}$, and we corrupt the triplet by replacing the tail user with probability $\frac{hpt}{tph+hpt}$.

4.4 Related Work

4.4.1 Network Embedding Models

There has been increasing attention on low-dimensional graph embedding recently. Many approaches have been proposed for data visualization, node classification, link prediction, and recommendation. DeepWalk [2] predicts the local neighborhood of nodes embeddings to learn graph embedding. LINE [3] learns feature representations in first-order proximity and second-order proximity respectively. GraRep [4] learns graph representation by optimizing k-step loss functions. Node2Vec [5] extends DeepWalk with a more sophisticated random walk procedure and explores diverse neighborhoods. Although many studies have reported

their performance on social network datasets, we argue that the actual social networks are more complicated. Users in social networks could have different neighbor structures based on different relationships. Jointly learning representations for users and relationships can help to describe users in social networks more precisely.

4.4.2 Structural Knowledge Graph Embedding Models

The main stream of structural embedding models follows the basic idea that every relation is regarded as translation in the embedding space. The embedding of one entity, say h, is close to another embedding, say t, by adding a relation vector r. A triplet (h, r, t) could be described as the equation $h + r \approx t$. TransE [7] first adopts the concept to learn entity embeddings in knowledge bases (KBs). However, TransE does not perform well on relations with reflexive (i.e., r is a reflexive map for triplets (h, r, t) and (t, r, h)), 1-to-N, N-to-1, and N-to-N properties. TransH [8] addresses this issue by introducing relationship hyperplanes so entities can be represented differently with respect to different relations. TransR [9] considers that entities and relations should be projected into different embedding spaces and mapped together by mapping matrices of relations. TransD [10] extends TransR but reduces its complexity by constructing two dynamic mapping matrices for each triplet and replacing matrix-vector multiplication operations by vector operations. Structural embedding models perform well for entity embedding in KBs, however, they only consider the network structure of entities—they do not use any information about textual communication. Since our study focuses on relationship and user embeddings in social networks, we conjecture that the textual communication between users plays an especially crucial role.

4.4.3 Text-aware Knowledge Graph Embedding Models

Some studies have introduced text-aware embeddings, which attempt to represent the knowledge graph with textual information. DKRL [12] proposes an encoder architecture with continuous bag of words (CBOW) and convolutional neural network (CNN) to learn entity embeddings based on network structure and entity description. SSP [13] introduces semantic hyperplanes to capture semantic relevance and correlate entity descriptions to certain topics.

These models perform well on knowledge graph embeddings, however, they only consider the description of entities as their textual information. Since our goal is to leverage the impact of interaction and communication between users in social networks, the user description does not provide sufficient details to describe these relationships between users. TransRev [88] learns a text representation for each pair of heterogeneous source and target nodes. Unlike knowledge graph models, it learns the "relationship" (i.e., textual review representation) between every user-product pair rather than a global relationship representation. Thus, the relationship learned from TransRev is incapable of modeling multiple relationships between a node pair like our proposed model.

Specifically, existing representation learning models for knowledge graphs only consider the information of each entity itself and then build a translative bridge to interpret the relation of two entities. As such, applying the existing models directly to social networks will disregard meaningful information because textual interactions between users can be important signals of the relationship of users. For example, messages between users suggest the topics they have in common. Those interactions enable us to estimate the strength of relationships and to further identify specific types of relationships among users. It facilitates a more accurate learning of hidden representations in social networks.

4.4.4 Comparison of TransConv to Related Work

To highlight differences with prior work, we list the score functions of related models in Table 4.1. The embeddings of user u_{i} and u_{j} are represented by vectors $\hat{u}_{\mathrm{i}}$ and $\hat{u}_{\mathrm{j}} \in \mathbb{R}^k$. In contrast with these models, which do not include textual communication in their score functions, we use the proposed conversation factors to augment the *TransConv* objective.

4.5 Experiments

We evaluate our approach and related methods on three various tasks: social network completion, triplets classification and multilabel classification.

Table 4.1. Score functions of embedding models

Model	**Score function** $f_r(u_{\mathrm{i}}, u_{\mathrm{j}})$
TransE	$\|\|\hat{u}_{\mathrm{i}} + \hat{r} - \hat{u}_{\mathrm{j}}\|\|_{l_{1/2}}; \hat{r} \in \mathbb{R}^k$
TransH	$\|\|(\hat{u}_{\mathrm{i}} - w_r^T \hat{u}_{\mathrm{i}} w_r) + \hat{r} - (\hat{u}_{\mathrm{j}} - w_r^T \hat{u}_{\mathrm{j}} w_r)\|\|_{l_{1/2}}; w_r, \hat{r} \in \mathbb{R}^k$
TransR	$\|\|M_r \hat{u}_{\mathrm{i}} + \hat{r} - M_r \hat{u}_{\mathrm{j}}\|\|_{l_{1/2}}; M_r \in \mathbb{R}^{n \times k}; \hat{r} \in \mathbb{R}^n$
TransD	$-\|\|M_{u_{\mathrm{i}}r} \hat{u}_{\mathrm{i}} + \hat{r} - M_{u_{\mathrm{j}}r} \hat{u}_{\mathrm{j}}\|\|_{l_{1/2}}; M_{u_{\mathrm{i}}r}, M_{u_{\mathrm{j}}r} \in \mathbb{R}^{n \times k}; \hat{r} \in \mathbb{R}^n$
DKRL	$\|\|\hat{u}_{\mathrm{i}} + \hat{r} - \hat{u}_{\mathrm{j}}\|\|_{l_{1/2}} + \|\|\hat{d}_{\mathrm{i}} + \hat{r} - \hat{d}_{\mathrm{j}}\|\|_{l_{1/2}} + \|\|\hat{d}_{\mathrm{i}} + \hat{r} - \hat{u}_{\mathrm{j}}\|\|_{l_{1/2}} + \|\|\hat{u}_{\mathrm{i}} + \hat{r} - \hat{d}_{\mathrm{j}}\|\|_{l_{1/2}}; \hat{r} \in \mathbb{R}^k$
TransConv	$[1 + \alpha\mu_{\mathrm{ij}}^r + (1-\alpha)\phi_{\mathrm{ij}}^r] \cdot \|\|w_r^T \hat{u}_{\mathrm{i}} w_r + \hat{r} - w_r^T \hat{u}_{\mathrm{j}} w_r\|\|_{l_{1/2}}; w_r, \hat{r} \in \mathbb{R}^k$

4.5.1 Datasets

We analyze two social network datasets in our experiments:

- The public Purdue Facebook network data from March 2007 to March 2008, which includes 3 million post activities. There are 211,166 triplets with 19,409 users. For every triplet $(u_{\mathrm{i}}, r, u_{\mathrm{j}})$, u_{i} posts at least one message (conversation) on u_{j}'s timeline and vice versa. We construct 41 relationships from user attributes, groups and top friends information.

- Our Twitter dataset is sampled from the dataset collected by [89]. It contains 20 million post activities from June to July 2009. There are 300,985 triplets with 22,729 users. We use the posts with user mentions (e.g., "@david happy birthday!") as textual interactions. The 42 relationships types are constructed from user profiles and follower/following information.

We follow TransE [7] to categorize relationships into four categories. In the Facebook (Twitter) dataset, there are 10.6% (23.6%) 1-to-1, 2.6% (6.6%) 1-to-N, 2.6% (6.6%) N-to-1 and 84.2% (63.2%) N-to-N relationships in generated triplets. Table 4.2 reports the statistics of two datasets. Compared to knowledge base datasets, our datasets is more challenging since it contains more N-to-N complex relationships. Table 4.3 lists the top-3 most frequent and bottom-3 least frequent relationships from the overall set of 41 (Facebook) and 42 (Twitter).

Overall, relationships with more examples have more textual conversations associated with them.

Table 4.2. Statistics of datasets

Dataset	**#User**	**#Rel**	**#Train**	**#Valid**	**#Test**
Facebook	19,409	41	126,963	42,101	42,102
Twitter	22,729	42	180,606	60,189	60,190

Table 4.3. Most- and least-frequent relationships in Facebook and Twitter datasets

Relationship	**#Sample**	**#Conversation**
Facebook		
top-3		
gender-male	29,818	89,060
looking-for-friendship	24,522	94,231
interested-in-women	23,776	73,860
bottom-3		
religious-view-hindu	42	124
hometown-california	34	139
relationship-status-complicated	10	86
Twitter		
top-3		
unverified-account	38,332	133,604
is-followed-by	36,883	128,370
uploaded-profile-image	33,496	113,279
bottom-3		
language-italian	20	83
location-canada	8	24
language-indonesian	4	17

4.5.2 Experiment Settings

We evaluate *TransConv* compared to several knowledge graph embedding models: transE, transH, transR, transD, and DKRL. Both structural and text-aware embedding models are

included. We follow the details in the papers to implement these models, and compare the performance of the above models by applying them on our social network datasets.

Table 4.4. Evaluation results of link prediction on Facebook dataset

Model	Mean Rank		Mean Hits@N (%)							
	Raw	Filter	N=10		N=5		N=3		N=1	
			Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter
TransE	305	304	50.6	52.3	37.3	39.9	27.3	30.3	11.4	13.5
TransH	168	168	73.8	76.3	57.5	62.2	43.1	49.0	18.7	23.7
TransR	195	194	75.5	78.7	56.3	61.9	41.6	48.0	18.0	22.7
TransD	295	294	50.6	52.2	37.3	40.0	27.5	30.5	11.4	13.8
DKRL(CBOW)+TransE	5,579	5,577	5.5	6.7	3.4	3.9	2.3	2.3	0.9	1.1
TransConv	**36**	**35**	**83.5**	**86.9**	**63.0**	**68.8**	**46.5**	**53.0**	**20.0**	**24.8**

Table 4.5. Evaluation results of link prediction on Twitter dataset

Model	Mean Rank		Mean Hits@N (%)							
	Raw	Filter	N=10		N=5		N=3		N=1	
			Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter
TransE	203	201	47.6	49.1	39.2	42.0	32.7	36.3	18.8	24.2
TransH	33	32	90.6	92.4	84.3	89.8	76.4	86.9	49.6	74.0
TransR	23	21	93.8	96.2	86.5	92.3	77.9	87.7	51.3	72.1
TransD	199	197	48.2	49.8	39.9	42.6	33.5	37.2	19.5	25.2
DKRL(CBOW)+TransE	5,706	5,704	1.1	1.1	0.7	0.7	0.5	0.5	0.2	0.2
TransConv	**9**	**5**	**95.6**	**98.3**	**88.6**	**95.7**	**80.2**	**91.6**	**52.2**	**74.2**

As described in Table 4.2, we perform stratified sampling to split the dataset and use training set and validation set to select the best configurations. Next, we perform *10-fold* cross validation on all data and report the average results. In training *TransConv*, we perform grid search over learning rate R for SGD among {0.001, 0.005, 0.01}, the batch size B among {100, 500}, the number of training epochs T among {200, 500}, the margin γ among {0.5, 1.0, 1.5}, the embedding dimension k among {100, 200, 300}, the norm used in score function among {*L1*-norm, *L2*-norm}, the top-K TF-IDF among {100, 500, 1000, 1500, 2000, 2500} and the learning weight α for conversation factors between 0 and 1. For

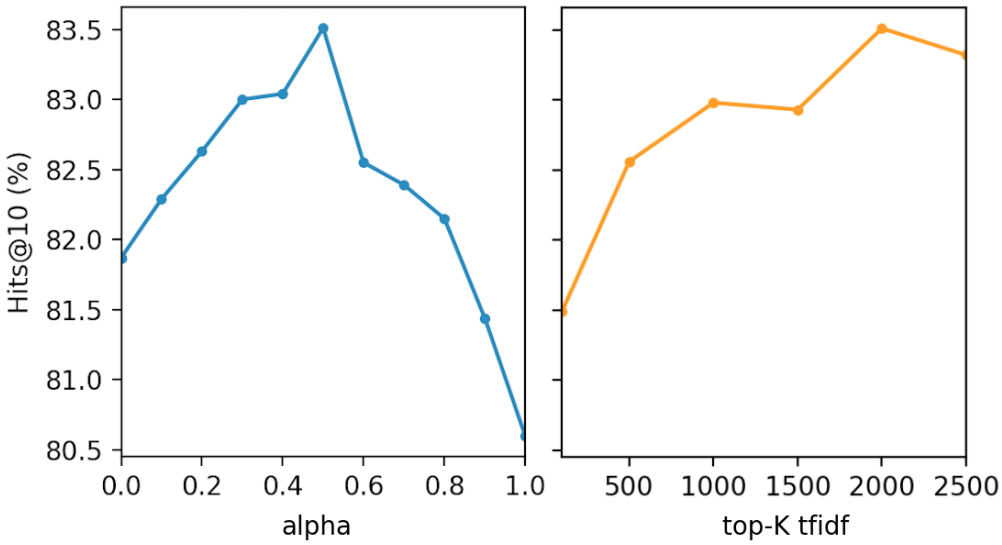


Figure 4.2. Sensitivity w.r.t. α and top-K TF-IDF on Facebook.

the Facebook dataset, the optimal configurations of *TransConv* are: $R = 0.001$, $B = 100$, $T = 500$, $\gamma = 1.0$, $k = 300$, norm = *L1*-norm, $K = 2000$ and $\alpha = 0.5$. The sensitivity of selecting α and top-K TF-IDF is reported in Figure 4.2. The best α we have is 0.5 and it suggests both conversation similarity and frequency factors take important roles in learning embeddings. The same configurations are applied to the Twitter dataset.

We follow the same process to select corresponding best configurations for other models. In training DKRL model, it is required to include textual information of each entity. In its original work [12], each entity's description is composed of a set of keywords selected from the entity's Wikipedia page. However, there is no direct textual description for Facebook and Twitter users. Therefore, we concatenate all the messages posted by a user as a document, and select keywords with top-K TF-IDF score to represent the user's textual description. We select $K = 1500$ for Facebook and $K = 2000$ for Twitter. Next, we apply Google's pre-trained Skip-Gram model [86], which is trained on part of Google News dataset (about 100 billion words), to generate each entity's description-based representation. Finally, we concatenate the learned description and structure-based representations for DKRL's prediction tasks.

Table 4.6. Detailed results by relationship categories with *Filter* setting on Facebook dataset

Model	Mean Rank				Mean Hits@10 (%)							
					Predicting *head user*				Predicting *tail user*			
Relationship Category	1-to-1	1-to-N	N-to-1	N-to-N	1-to-1	1-to-N	N-to-1	N-to-N	1-to-1	1-to-N	N-to-1	N-to-N
TransE	1054	199	165	216	21.6	54.1	56.5	55.8	23.5	54.1	60.9	55.9
TransH	1062	65	147	58	22.2	74.2	59.4	83.7	21.7	73.7	63.6	83.7
TransR	230	361	331	180	73.2	**81.5**	77.7	79.6	71.7	76.8	80.3	79.5
TransD	1023	188	202	208	23.3	56.1	48.6	56.7	23.3	51.6	54.3	54.9
DKRL(CBOW)+TransE	5,612	5,758	5,323	5,575	4.6	4.5	3.3	5.8	4.6	6.7	2.2	5.7
TransConv	**47**	**31**	**30**	**35**	**88.2**	79.7	**80.6**	**86.8**	**88.6**	**84.3**	**83.7**	**87.3**

Table 4.7. Detailed results by relationship categories with *Filter* setting on Twitter dataset

Model	Mean Rank				Mean Hits@10 (%)							
					Predicting *head user*				Predicting *tail user*			
Relationship Category	1-to-1	1-to-N	N-to-1	N-to-N	1-to-1	1-to-N	N-to-1	N-to-N	1-to-1	1-to-N	N-to-1	N-to-N
TransE	136	41	57	256	57.6	73.5	67.4	42.2	57.9	69.7	71.3	40.5
TransH	91	16	5	8	76.8	90.5	93.2	**98.4**	76.8	89.1	95.6	98.3
TransR	64	**3**	**3**	9	94.4	95.4	94.6	95.0	94.2	95.0	95.3	94.9
TransD	111	76	76	254	64.6	64.4	61.9	40.9	63.1	65.2	68.5	41.9
DKRL(CBOW)+TransE	5,588	5,796	5,511	5,758	1.1	0.6	1.0	1.0	1.4	1.1	1.1	1.1
TransConv	**6**	6	6	**4**	**98.1**	**98.3**	**98.0**	**98.4**	**98.2**	**98.1**	**97.6**	**98.5**

Figure 4.3. Evaluation results for link prediction on most frequent (top row) and least frequent (bottom row) relationships on Facebook dataset.

4.5.3 Social Network Completion

In this experiment, we evaluate whether the learned user and relationship embeddings are useful in predicting the existence of user pairs that actually have certain relationships.

The task is to complete a golden triplet (u_i, r, u_j) by minimizing the score function $f_r(u_i, u_j)$, as defined in Table 4.1, when u_i or u_j is missing. For example, we predict u_j given (u_i, r) or predict u_i given (r, u_j). We follow the same protocol used by TransE [7]. First, we compute the raw scores for those corrupted triplets and rank them in ascending order, then get the rank of the original golden triplet. Additionally, it is possible that a corrupted triplet exists in the graph and is ranked before the original triplet. This case should not be considered as wrong, so we also compute the "filter" scores to eliminate the factor. The mean rank of correct users and *Hits@N*, the proportion of correct users in top-N ranked users, are reported in Table 4.4 and Table 4.5. A lower mean rank is better while a higher

Hits@N is better. The results show that *TransConv* consistently outperformed other models and achieved 86.9% on Facebook dataset and 98.3% on Twitter dataset with filter setting in *Hits@10*. The results in bold statistically significant outperform other models at 0.01 level in paired t-tests.

First, it is interesting that *TransConv*, TransH, and TransR have the top-3 highest mean rank among models in both datasets, which shows projecting matrices to relation hyperplanes and spaces is effective when learning embeddings for social network data. Secondly, the performance difference between *TransConv* and TransH suggests that considering the text similarity and communication intensity between users improves the embedding learning significantly. Thirdly, as the reported results of different α values in Figure 4.2, it further indicates text similarity and communication intensity are complementary factors since neither $\alpha = 0$ nor $\alpha = 1$ achieve the best result. Furthermore, it is noticeable that DKRL model did not perform well with both datasets and that might be caused by the way how we generate the textual description for users. Unlike texts in Wikipedia page are used to define and describe an entity, the collected messages could be very casual, noisy and short of meaningful words to depict a user.

We further investigate the performance on each relationship category and report the results of mean rank and *Hits@10* in Table 4.6 and Table 4.7. In order to ascertain the consistency, we also evaluate the results of replacing head or tail users. In Figure 4.3, we examine more closely on the top-3 most frequent and bottom-3 least frequent relationships of the Facebook dataset. Generally speaking, all models have higher *Hits@10* scores and lower score variances on top-3 relationships. The results show all models perform stable on relationships that contain more samples of golden triplets. In top-3 relationships, *TransConv* and TransH outperform others and there is no significant performance difference between *TransConv* and TransH. It reconfirms that it is helpful to include relationship hyperplane projection for social network datasets. However, when examining the bottom-3 relationships, *TransConv* still achieves nearly over 60% in *Hits@10* and outperforms others; While the performances of other models, including TransH, have dropped significantly to lower than 20%. In addition, only *TransConv* and DKRL achieve over 10% in the bottom-1 relationship. It suggests that incorporating textual information is beneficial in learning social

relationship representation. We do not include the figure for Twitter dataset here due to space limitation, but the result is also consistent with the result on Facebook dataset. In overall, *TransConv* consistently performs better on both top-3 and bottom-3 relationships and shows more robustness for lack of training samples.

4.5.4 Triplets Classification

In this task, we evaluate if the score function of *TransConv* is effective in discriminating golden and negative triplets by binary classification. For a triplet (u_i, r, u_j), it is predicted as positive if its score $f_r(u_\mathrm{i}, u_\mathrm{j})$ is lower than the threshold σ_r; Otherwise, it is predicted as negative. The relation-specific threshold σ_r is determined by maximizing the classification accuracy on the validation set. It requires negative labels to perform the evaluation. We follow the same setting in TransE [7] to construct negative examples for Facebook dataset resulting with equal number of positive and negative examples, and we further discuss three negative sampling strategies by replacing head users, replacing tail users or randomly selecting head (tail) users to replace. When constructing a negative triplet, we constrain the replaced users by only allowing users in a position if they appeared in that position and was ever in that relationship with others in the dataset. For example, with the strategy of replacing tail users, given a correct triplet ($user_7$, $is\_top\_friend\_of$, $user_{30}$), a potential negative example is ($user_7$, $is\_top\_friend\_of$, $user_{15}$). The $user_{15}$ adds other users as his (her) top friends on Facebook (in the position of tail user) but not including $user_7$.

We compare the performance of knowledge graph and networking embedding models and report classification accuracy on eight selected relationships in Table 4.8. We first select top-5 most frequent relationships, which all happen to be N-to-N category in both Facebook and Twitter dataset. We also include the relationship with largest number of triplets in 1-to-1, 1-to-N, and N-to-1 category respectively to consider all relationship categories in our experiments. For knowledge graph models, different score functions as described in Table 4.1 are evaluated. For network embedding models, since they do not learn relation embedding, we concatenate the learned user embeddings of each pair $(u_\mathrm{i}, u_\mathrm{j})$ as a feature vector $\mathrm{e}_{u_\mathrm{i}} \oplus \mathrm{e}_{u_\mathrm{j}}$ and train a binomial logistic regression model for each relation r. That is, if $(u_\mathrm{i}, r, u_\mathrm{j})$ is a golden triplet, the label of input $\mathrm{e}_{u_\mathrm{i}} \oplus \mathrm{e}_{u_\mathrm{j}}$ is true, otherwise false. The results show the score

Table 4.8. Mean accuracy (%) for triplet binary classification on selected relationships with different negative sampling strategies

Dataset	Facebook			Twitter		
Negative Sampling	replace *head user*	replace *tail user*	replace *random*	replace *head user*	replace *tail user*	replace *random*
TransE	79.3	81.7	75.0	64.4	64.1	62.8
TransH	71.3	71.2	67.9	65.3	65.4	63.4
TransR	91.7	91.6	82.7	80.6	80.5	78.5
TransD	67.0	67.1	63.6	63.9	63.8	62.8
DKRL(CBOW)+TransE	50.3	50.3	52.1	50.0	50.0	44.0
Node2Vec	74.7	74.2	75.1	66.6	68.3	65.8
LINE(1st+2nd)	75.1	73.5	76.1	65.8	64.8	65.8
TransConv	**94.9**	**94.9**	**83.5**	**99.9**	**99.9**	**88.2**

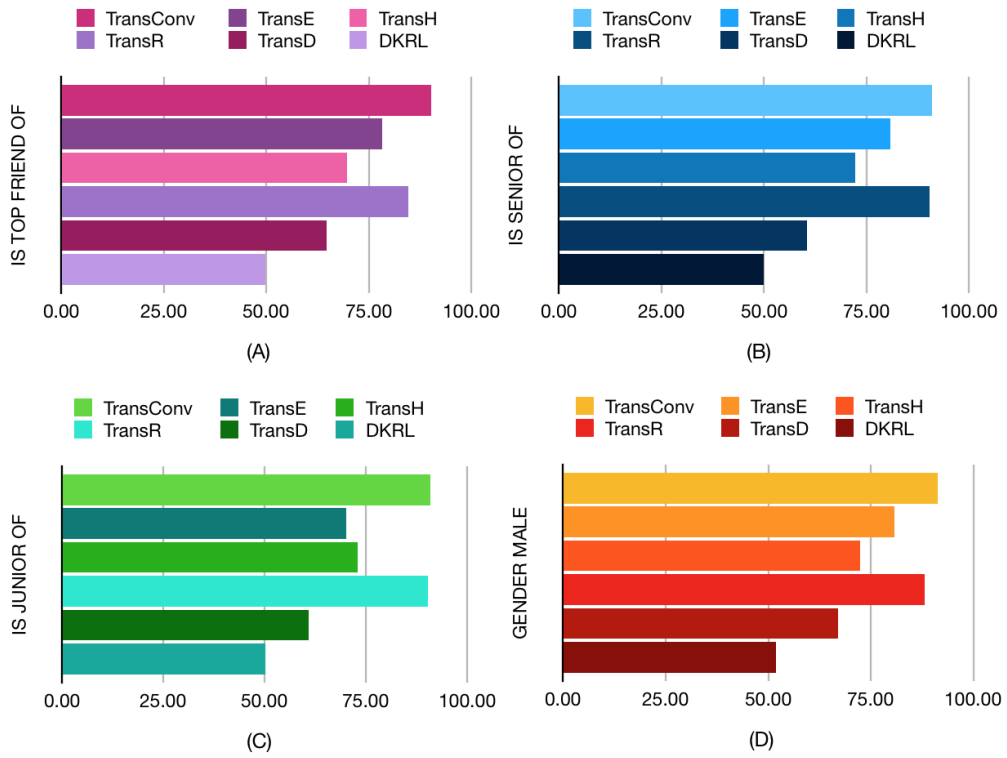


Figure 4.4. Results of triplet classification on different relationship categories of Facebook dataset. The relationship in (A) belongs 1-to-1, (B) belongs to 1-to-N, (C) belongs to N-to-1, and (D) belongs to N-to-N category.

function of *TransConv* significantly outperforms other models in triplets binary classification. With incorporating conversation factors, it enables our proposed score function to identify golden/negative triplets in more precise way. In addition, we evaluate the results on different relationship categories. In Figure 4.4, we show the four relationships of Facebook dataset that each one contains the largest number of triplets in its own category. *TransConv* also outperforms other models in all four categories.

4.5.5 Multilabel Classification

In this section, we evaluate if the representations learned from *TransConv* is effective for multilabel classification on the relationship labels of user pairs. We use the same relationships selected in the experiments of triplet classification.

However, since the user representations learned from knowledge graph embedding models vary from different relationships, the common one-vs-all approach [90] for multilabel classi-

Table 4.9. Results of multilabel 8-relationship classification

Model	Hamming Score - Accuracy (%)	
Dataset	Facebook	Twitter
TransE	12.4	37.7
TransH	13.8	37.8
TransR	14.0	38.1
TransD	10.8	37.4
DKRL(CBOW)+TransE	7.1	39.3
TransConv	**15.2**	**39.6**

fication is not applicable. We design a multilabel classification experiment based on global score threshold σ learned from the validation set. The experiment is constructed as below:

1. For each user pair $(u_{\mathrm{i}}, u_{\mathrm{j}})$ in the validation set, we retrieve the scores $f_r(u_{\mathrm{i}}, u_{\mathrm{j}})$ on every relation r and further normalize the scores by z-score.

2. For each embedding model, we search a global score threshold σ among all relations and use it in prediction task. That is, if the normalized score of a user pair $(u_{\mathrm{i}}, u_{\mathrm{j}})$ in any relation r is smaller than σ, we predict $(u_{\mathrm{i}}, r, u_{\mathrm{j}})$ as a true triplet; otherwise, predict it as a negative one. Predicting a triplet as true means predicting the user pair hold the relationship label. It accordingly predicts each user pair a set of relationship labels.

3. We perform an exhaustive search for the global threshold σ that achieves the highest *hamming score* [91] on validation set. Let T be the true set of labels and S be the predicted set of labels. Accuracy is measured by hamming score which symmetrically measures how close T is to S, defined as $Accuracy = |T \cap S|/|T \cup S|$.

4. We follow the same steps to get the normalized scores for every triplet in the testing set and predict the relationship labels by the global threshold σ. Then we report the hamming score as the testing accuracy of multilabel classification.

We apply the same experiment procedure for all models and the results are shown in Table 4.9. *TransConv* performs best in 8-relationship classification task. It shows that taking

our proposed conversation factors into consideration is effective to capture the strength of relationship.

4.6 Discussion

Relational representation learning for social networks facilitates real-world tasks such as visualization, link prediction and friend recommendation. Traditional knowledge graph embedding models learn continuous low-dimensional embedding of entities and relations. However, when applied to social networks, existing approaches do not consider the rich textual communications between users, which contains valuable information to describe social relationships. The main contribution in this work is to propose a novel approach, *TransConv*, that incorporates textual interactions between pair of users to improve representation learning of both users and relationships. Our experiments on real social network datasets show *TransConv* approach learns better user and relationship embedding compared to other state-of-the-art knowledge graph embedding models. Moreover, the results illustrate that our model is more *robust* for sparse relationships where there are fewer examples.

5. JOINT EMBEDDING MODELS FOR TEXTUAL AND SOCIAL ANALYSIS

5.1 Introduction

The remarkable popularity of social media outlets provides exciting opportunities to study language and social behavior on a large scale. These outlets allow users to openly interact, share content, endorse and disapprove of the behavior and stances of each other. The interconnected structure of this data strongly suggests that it should be studied in a holistic way, taking into account the connections between the social behavior of users, their stances and viewpoints, and the content generated when they interact with other users. Taking this approach would assist social network researchers to understand the communication patterns between users in the network. In addition, it would also allow us to study natural language in a holistic way, taking into account the social context in which it was generated.

Unfortunately, most works fall into two, almost completely disconnected, camps. The first focuses on *social network analysis* and looks at the network structure and information flow on it as means of inferring knowledge about the network, while the other focuses on *natural language analysis*, looking into tasks such as extracting social relationships from narrative text [14]–[16] and analyzing the contents of the information flowing through the network. Both perspectives resulted in a wide range of successful applications; however, they neglect to model the interactions between the social and linguistic representations and how they complement one another. A few recent exceptions look at the connections between the two. In [17] the network structure is represented explicitly. The authors infer sentiment links between nodes in a social network by combining local sentiment analysis decisions with global inference over the network structure. In [18] the network structure is represented by embedding users in a dense low dimensional space based on their social connections. In both of these works, there is a clear divide between the social and language analysis. The social aspect is modeled separately and then used to improve the inferred predictions or disambiguate language use.

In this work we suggest a holistic approach, combining users' information, their social behavior and language use, into a joint model. Unlike previous works, which model the

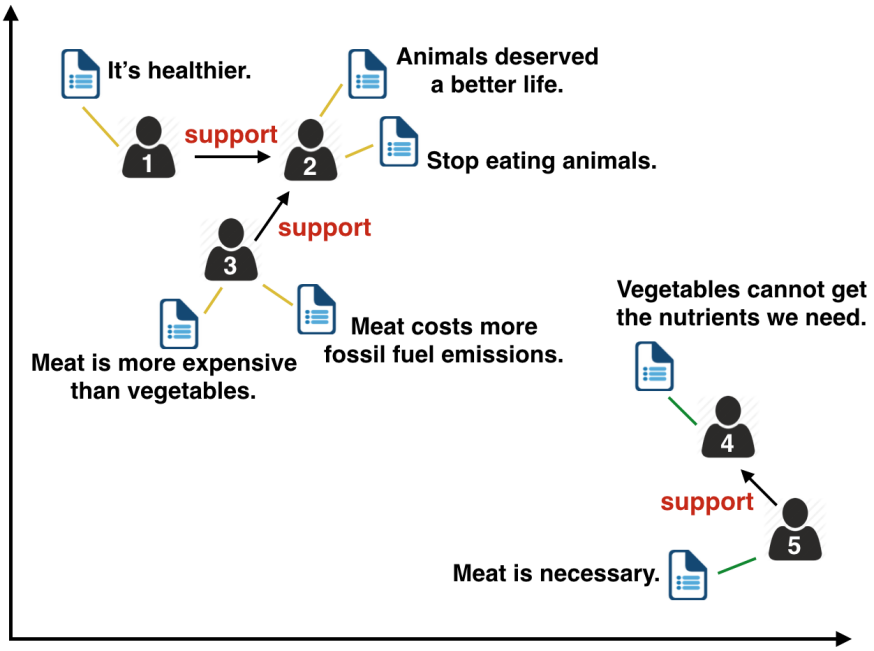


Figure 5.1. Illustration for proposed model on Debate dataset.

connections between these aspects either by using collective classifications approaches *or* by creating user embeddings based on their social structure, in this work we suggest that the two can be combined, and show empirically the advantage of this combination.

The broadly applicable settings of our model can be represented as a graph, connecting users with one another using edges that capture social behaviors, and users with text, which is reflective of their opinions, age (and other attributes) and social goals. For example, the graph illustrated in Figure 5.1 describes a debate scenario, in which *user* 2 and *user* 4 are the debaters, and *user* 1 and *user* 3 support one side, while *user* 5 supports the other. Our model uses the structured information in the graph in two ways.

5.1.1 Joint Embedding

In the first step, we embed users and their textual content in a single vector space. This representation utilizes the social signal represented in the graph edges. In addition, it also incorporates different types of social signals between users, such as the text of messages they post, and the attributes they share. As illustrated in Figure 5.1, users' stances (which are part of the user attributes we model) can be reflected in the way they communicate and

the textual content they generate. We view this correspondence as a form of alignment, allowing us to identify similar attributes between users. Our social network embedding approach is different than previous work [2], [3], [5], which treats graph nodes as discrete objects whose embedding is only determined by the edges that connect them to other nodes. Instead in our model, the node embedding is determined both by the graph structure and the attributes associated with each node. This approach allows us to generalize better, as it identifies the similarity between nodes that share attributes even if their social connections are *sparse*. We demonstrate this point empirically, showing that our model can outperform the state-of-the-art Node2Vec model.

Our approach represents text and users in the same space. As a result, the embedding of textual content captures the ideological stances of its author. To the best of our knowledge, up to the time, none of studies addressed these viewpoints.

5.1.2 Joint Prediction

Representing users and textual content in the same space allows us to reason about the relationships between the users and the content generated by them. We define multiple decision tasks over these relationships and show how these prediction can be defined over the embedding space by comparing the similarities of pairs of vectors representing objects. In the second step of our model, we exploit the graph structure to perform collective classification to ensure *consistency* between these decisions. For example, in Figure 5.1, *user* 1 supports *user* 2. Identifying this relationship can inform us about the relationship between *user* 1 and the content generated by *user* 2.

We evaluated our approach empirically over two very different datasets, extracted from a debate forum *debate.org*, and social networking site *facebook.com*. Our results consistently show that jointly embedding users and language results in better performing models, and using the dependencies defined by the graph structure also at prediction time can lead to even further improvements.

5.2 Related Work

Several recent works have explored embedding methods for social network analysis. The graph embedding approach suggested by [92] constructs a low-dimensional embedding of graphs through matrix factorization. Other works focus on social networks directly, [2] considers the signal of network structure by either observing direct linkage information or exploring neighbors over random walks. [3], [5] are inspired by Skip-gram model. Learning network embedding through exploring the neighborhood (contexts) in a BFS, DFS, or hybrid fashion.

Several works have also looked into social representation for NLP problems. For example, [55] first generate several different representations via text or network structure as views, then combine these views into a single embedding. [18] learn community-specific projections of word embeddings, assuming that word will have similar sentiment meaning for their near neighbors.

Our work explores the connections between text and users attributes, attempting to create a common representation for the two. [93], [94] learns joint embeddings of images and text in a deep neural network framework, which resembles our work.

There is little work looking into a shared model connecting text, attributes and behavior. The closest to our work is [54], that jointly integrates different kinds of cues (text, attribute, graph) into a single latent representation to get user embeddings.

5.3 Learning

Our first step towards jointly modeling social interactions and language is to learn a joint embedding of user characteristics and language. From previous works on node/graph embedding, it is clear that capturing the dependency between different aspects of users, namely text, attributes, graph structure (e.g. friends), user behaviors (e.g. votes) etc., may help improve predictions.

Inspired by the works in computer vision, which maps text and images into the same space [93], we follow the intuition that different aspects of users can be mapped into the same space so that we may infer knowledge about one aspect from the other. For example,

we can align text with the attributes of a user. Our embedding method is similar in spirit to the Deep Structured Semantic Model (DSSM) [95], allowing us to infer the most probable attributes given the text explicitly or implicitly using the trained model.

5.3.1 Joint Embedding

Let U denote the set of all users, let A and T denote the set of all attribute vectors and text authored by those users respectively. Our objective is to learn a semantic embedding for both users text and user attributes so that they are close in the embedding space if they are semantically close to each other. For a text (or attribute) input x, we will compute its embedding e using M hidden layers $l_{\mathrm{i}}, \mathrm{i} = [0, M-1]$. The first hidden layer l_1 is computed from the input x:

$$l_0 = f(W_{\mathrm{i}} x_{\mathrm{i}} + b_{\mathrm{i}})$$

Subsequent layers are computed recursively:

$$l_{\mathrm{i}} = f(W_{\mathrm{i}} l_{\mathrm{i}-1} + b_{\mathrm{i}}), \mathrm{i} = 1, ..., M-1$$

Then the final layer produces the embedding:

$$\mathrm{e} = f(W_M l_{M-1} + b_M)$$

To learn the embedding, we will divide a large social network into small sub-networks. These sub-networks capture meaningful scenarios (e.g., a debate between users, and subsequent voting by other users), and are used to construct the embedding objective function and constrain the decision space at prediction time. All training and test examples are generated from within each sub-network.

For the debate dataset, it is natural to consider a debate between two participants as a sub-network. Hence, we have the two users (which we refer to as **author**), arguments from multiple rounds in the debate (which we refer to as **text**), and other users who voted for either side (which we refer to as **endorser**) in the sub-network.

For the Facebook dataset, we randomly divide all users into pairs. We extract a sub-network that contains a pair of users (authors), posts generated by each of them (text), and other users who set either of them as top friends (endorser). Note that in order to control the size of the problem, we consider at most 5 posts for each user.

5.3.2 Embedding Views

Given the scenario described above, we can now describe several different objectives that we can consider while learning the embedding. Each of them focuses on two contending relations (edges in the graph).

Text vs. Attribute (TA)

This objective is to distinguish text t_{i} written by an author u_{i} from text t_{j} that is written by another user $u_{\mathrm{j}\neq\mathrm{i}}$, by using the attribute representation of u_{i} (i.e., based on a_{i}). For the debate dataset, positive examples consist of $(a_{\mathrm{i}}, t_{\mathrm{i}})$ pairs (attributes and text from one round of the debate of user u_{i}). Negative examples consist of $(a_{\mathrm{i}}, t_{\mathrm{j}})$ pairs, where t_{j} is the text in the same round but from the contender in the debate. In the Facebook dataset, positive examples are constructed from user wall posts, negative examples use a wall posting from a random non-friend.

Text vs. Text (TT)

This objective is to distinguish text t_{i}^{a} written by an author u_{i} from text t_{j} that is written by another user $u_{\mathrm{j}\neq\mathrm{i}}$, by using the language representation of u_{i} ($t_{\mathrm{i}}^{b\neq a}$). We consider adjacent debate rounds or posts (e.g., t_{i}^{b}, t_{i}^{a}) as positive examples for this objective. For negative examples, we use text from the contender (e.g., t_{i}^{b}, t_{j}^{c}).

Attribute vs. Attribute (AA)

This objective is to distinguish author u_{j} supported by endorser u_{i} from the author u_k that is not, using the attribute representation of each user (e.g., a_{i}). For the debate dataset, positive examples consist of $(a_{\mathrm{i}}, a_{\mathrm{j}})$ pairs where u_{j} voted for u_{i} in a debate. Negative

examples consist of (a_{i}, a_k) pairs, where u_k voted against u_{j}. In the Facebook data, positive examples are constructed from top-friend links and negative examples are constructed from non-friends.

5.3.3 Embedding Objective

Given two pairs of objects, one pair o, c^p corresponding to a positive example and one pair o, c^n corresponding to a negative example (as defined by the embedding views), we can define our embedding loss for each view —

$$L_V = \sum_{(o_{\mathrm{i}}, c^p_{\mathrm{i}}, c^n_{\mathrm{j}})} l(sim(\mathrm{e}_{o_{\mathrm{i}}}, \mathrm{e}_{c^p_{\mathrm{i}}}), sim(\mathrm{e}_{o_{\mathrm{i}}}, \mathrm{e}_{c^n_{\mathrm{j}}})) \tag{5.1}$$

In the Text vs. Attribute (TA) view, o corresponds to a user's attribute vector (a), and c^p, c^n correspond to texts t^p, t^n written by the user and by a different user, respectively.
We consider the similarity among the embeddings $\mathrm{e}_{o_{\mathrm{i}}}$, $\mathrm{e}_{c^p_{\mathrm{i}}}$, and $\mathrm{e}_{c^n_{\mathrm{j}}}$. The goal is to maximize the similarity between the user's attribute and textual representation, while minimizing the similarity between the user's attribute representation with the textual representation of another user. $sim()$ is a vector similarity function; we use cosine in this work. $l()$ is the cross-entropy loss.

$$l(p, n) = -\log\left(\frac{\mathrm{e}^p}{\mathrm{e}^p + \mathrm{e}^n}\right)$$

Joint Embedding Loss Function

We combine the embedding objective for each view into a joint training objective:

$$L(U, T, A) = L_{TA} + \lambda_1 L_{TT} + \lambda_2 L_{AA} \tag{5.2}$$

Similar to the Structure-preserving constraints proposed in [94], we also take advantage of these implicit constraints in our joint embedding model. We then use mini-batch gradient descent to minimize the loss.

5.4 Prediction

Our joint embedding model maps users (represented as attributes vectors) and text into the same space. This mapping reflects the users attributes, the relationships between them and the content they authored. Thus allowing us to compute the similarity between any pair of users, texts or their combination. This is a very useful property, as multiple prediction tasks can be defined over this similarity metric without requiring further retraining, simply by defining these tasks as "multiple choice" predictions which can be decided by comparing the similarity scores of candidate pairs.

In this work we consider four such tasks, capturing relationships between users and text. These prediction tasks are highly inter-dependent. For example, if we know two texts are written by the same author, then identifying the author of the first piece of text also determines the authorship of the other one. We exploit these dependencies by making a joint prediction over multiple instances. We formulate the decision as an Integer Linear Program (ILP) which allows us to directly force the consistency between decisions. We define the ILP objective function using the similarity scores between objects, defined by the joint embedding space.

5.4.1 Prediction Tasks

We define multiple prediction tasks, each requiring the model to decide between two alternatives. In all of these decisions, one of the candidates will result in a correct decision.

AuthoredBy

Given a user u, and two text candidates t_p and t_n, predict which one is authored by u.

SameAuthor

Given three text candidates t_i, t_j and t_k, predict which of the latter two is from the same user as t_i.

AgreeWith

Given three users u_{i}, u_{j} and u_k, predict which of the latter two u_{i} agrees with. This task is defined over the *VoteFor* relation in the debate dataset, and over the *TopFriend* relation in the facebook dataset.

SupportedBy

Given one user v, and two text t_p and t_n, predict which text user v supports. This task is defined over the *EndorsedBy* relation in the facebook dataset.

Similar in spirit to [96], we define an instance of the above four tasks as the triad $(x_{\mathrm{i}}, x_{\mathrm{j}}, x_k)$ of items where x_{i} is called the probe item and x_{j} and x_k are called the test items. We can make the decision locally by comparing $sim(\mathrm{e}_{x_{\mathrm{i}}}, \mathrm{e}_{x_{\mathrm{j}}})$ and $sim(\mathrm{e}_{x_{\mathrm{i}}}, \mathrm{e}_{x_k})$, and predicting the relation holds for the semantically closer pair.

5.4.2 Joint Prediction

We model the dependencies between decisions by predicting them jointly. To do so, we will define the predictions over the sub-network of any two given users as an Integer Linear Programming (ILP) instance.

The ILP global optimization goal is defined over two given users u_{i}, u_{j}, the textual content generated by the two users, $\{t^0_{\mathrm{i}}, ..., t^k_{\mathrm{i}}\}$, $\{t^0_{\mathrm{j}}, ..., t^k_{\mathrm{j}}\}$, generated by u_{i}, u_{j} respectively, and other users $\{u^0_{\mathrm{i}}, ..., u^m_{\mathrm{i}}\}$, $\{u^0_{\mathrm{j}}, ..., u^m_{\mathrm{j}}\}$, who have *supported* u_{i}, u_{j} respectively.

We create four types of boolean decision variables corresponding to the four decision tasks above. Specifically, we associate a boolean variable $\alpha_{k,l}$ with each one of the users ($k = \{\mathrm{i}, \mathrm{j}\}$), and the text t^l_k, and associate a score $sim(\mathrm{e}_{u_k}, \mathrm{e}_{t^l_k})$ with that variable. Similarly, we associate a boolean variable $\beta_{\mathrm{i,j}}$ with every two texts, and associate a score $sim(\mathrm{e}_{t_{\mathrm{i}}}, \mathrm{e}_{t_{\mathrm{j}}})$ with it. Another boolean variable $\gamma_{k,l}$, for any two users u_k, u_l, who have *supported* either u_{i} or u_{j}, and associate a score $sim(\mathrm{e}_{u_k}, \mathrm{e}_{u_l})$ with it. Finally, we associate a boolean variable $\delta_{k,l}$, with pairs consisting of a user u_k, who has *supported* either u_{i} or, u_{j}, and text t_l, and associate a score $sim(\mathrm{e}_{u_k}, \mathrm{e}_{t_l})$ with it. The set of all possible decisions for the tasks are

denoted as A for the AuthoredBy task, B for the SameAuthor task, Γ for the AgreeWith task and Δ for the SupportedBy task. Given these variables, our prediction function is:

$$\arg\max_{\alpha,\beta,\gamma,\delta} \sum_{\alpha\in A} \alpha \cdot score(\alpha) + \sum_{\beta\in B} \beta \cdot score(\beta) + \sum_{\gamma\in\Gamma} \gamma \cdot score(\gamma) + \sum_{\delta\in\Delta} \delta \cdot score(\delta)$$

Subject To **C**

where **C** is a set of constraints defined as follows:

Intra-task constraints:

We define two types of intra-task constraints that restrict the decisions within one task.

1. Given two users u_1, u_2 and text t from one of them:

$$\texttt{AuthoredBy}(t_1,\ u_1) + \texttt{AuthoredBy}(t_1,\ u_2) = 1$$

2. Given a pair of texts with different authors, t_2 and t_3, and another text t_1 sharing an author with either t_2 or t_3:

$$\texttt{SameAuthor}(t_1,\ t_2) + \texttt{SameAuthor}(t_1,\ t_3) = 1$$

Inter-task constraints:

These are constraints that require decisions between multiple task to be consistent.

3. Given two texts t_1, t_2, and user u:

$$\texttt{SameAuthor}(t_1,\ t_2) \wedge \texttt{AuthoredBy}(t_1,\ u) \Rightarrow \texttt{AuthoredBy}(t_2,\ u)$$

4. Given two users u, v, and text t:

$$\texttt{AgreeWith}(v,\ u) \wedge \texttt{AuthoredBy}(t,\ u) \Rightarrow \texttt{SupportedBy}(t,\ v)$$

Table 5.1. Number of examples for each relation type

Relation	Debate	Facebook
AuthoredBy	120264	54800
SameAuthor	90720	27400
AgreeWith	17398	2958
SupportedBy	45398	11796

We refer to the model using intra-task constraints only as a *semi-joint* model, and *joint* refers to the full set.

5.5 Empirical Evaluation

We evaluate the capability of our embedding model to reconstruct all relations between text and users in the social network. We have four prediction tasks, each of them focuses on one type of relation in the graph. We run experiments on the following two datasets.

5.5.1 Datasets

Debate.org

This is a dataset crawled by us from the debate.org website in June 2016. In contains user attributes (e.g., age, gender, etc.) and stances on controversial issues, the debate text and voting behavior by non-participating users, which determine who wins the debate. For this experiment, we only kept users whose attributes are not empty, and debates between these users. It resulted in 13,268 users, 23,585 debates and 60,132 debate rounds.

Purdue Facebook Network

The Purdue Facebook network data includes three million public post activities created by Purdue students from March 2007 to March 2008. After filtering out posts with only few words and users with no sufficient profile information, 14,804 users and 207,596 post activ-

ities are used in our experiments. We collected top-friend links and attributes from profile (hometown, interested_in, looking_for, political_view, relationship_status, religious_view, sex, group, network) for each user.

The statistic about number of examples for each relation type is shown in Table 5.1.

5.5.2 Experimental Settings

We used Theano [97] to implement the embedding neural network and Gurobi [98] as our ILP solver. The embedding neural network for both text and users contain two hidden layers and 300 hidden units in each layer. Hyperbolic tangent (tanh) is used as non-linear activation function. The vector comes out of the last hidden layer is considered as the embedding. For training, we used mini-batch gradient descent with early stopping based on the development set. All experiments use 5-fold cross validation, using one fold for testing, one fold for development and three for training.

Input Representation

The initial input representation, for the text and the user attributes, over which the embedding function is defined, is an important consideration. For the former, it is natural to use average word embedding as the text input. We also explored two alternatives of text input in our evaluation.

- **Word Hashing** This approach proposed in [95], converts text into a lower dimension vector by extracting letter n-gram as features. This method can dramatically reduce the representation dimensionality compared to the *one-hot* representation (i.e., unigram features) with a negligible collision rate. Since it uses letter n-gram, it can also capture morphological variations of words and generate features for unseen words in the training set.

- **Skip-Thought Vectors** [99] extends the idea of skip-gram in training word embedding to sentence embedding learning. They trained an encoder-decoder model based on the continuity of text from books. Such a method would take both semantic and syntactic properties of a sentence into account when generating text embedding.

There are also multiple ways to represent users in a social network. For example, the intrinsic attributes listed in one's profile or the stance one holds on certain topics. We explored three types of attribute input:

- **Attributes** One hot representation of the user's attribute values.
- **Attributes Words** Since the one-hot attribute vector is sparse, we also extracted the average embedding of those attribute words (e.g., "female", "graduate degree") as features.
- **Stance** On the debate dataset, we can also make use of the stance on big issues (e.g. abortion, gun control) as a user representation.

In our experiments, we used average word embedding as the text input for both datasets. The concatenation of one-hot attribute vector and attribute words are regarded as user input for the facebook data. We observed how different initial representations can impact the embedding, these are summarized in Table 5.2.

Table 5.2. Results on AuthoredBy task using different texts and user inputs on Debate dataset

text input	user input	accuracy
WE	attribute	56.39
WE	+stance	62.16
WE	+stance, words	65.53
word hashing	+stance, words	66.45
skip-thought	+stance, words	67.97

5.5.3 Results

We show the accuracy on test set under different training and prediction methods in Table 5.3 for the debate data, and Table 5.4 for the facebook data. The reported results

Table 5.3. Accuracy of prediction tasks under different settings on Debate dataset

training method	prediction method	*prediction task* AuthoredBy	SameAuthor	VoteFor	SupportedBy	average
WE-baseline	local	50.99	61.62	57.88	52.27	55.69 (55.16)
LR-baseline	local	62.18	50.00	54.94	51.15	54.57 (55.85)
Node2Vec	local			60.09		
L_{TA}	local	65.53	64.74	58.08	53.51	60.47 (62.80)
L_{TT}		50.32	66.73	61.78	50.48	57.33 (56.51)
L_{AA}		49.90	61.21	65.50	49.00	56.40 (54.49)
$L(joint)$		64.08	66.68	63.64	55.28	62.42 (63.45)
L_{TA}	semi-joint	70.13	67.65	58.08	53.51	62.34 (65.79)
L_{TT}		50.68	70.48	61.78	50.48	58.36 (57.91)
L_{AA}		50.15	62.86	65.50	49.00	56.88 (55.15)
$L(joint)$		69.30	70.15	63.64	55.28	64.59 (66.90)
L_{TA}	joint	73.09	86.97	58.35	55.26	68.42 (73.80)
L_{TT}		50.77	88.81	60.99	50.86	62.86 (64.04)
L_{AA}		49.58	85.52	65.49	49.04	62.41 (62.41)
$L(joint)$		72.64	88.62	62.84	58.16	70.57 (74.91)

are averaged across 5 folds. We compare several training methods (first column). Our WE-baseline (SE-baseline) model is to simply use the average word (sentence) embedding to represent both text and attributes. LR-baseline model utilize logistic regression. It uses the concatenation of both input vectors as features, and consider the positive pairs as positive examples and vice versa. For example, given user a with attribute $attr_a$, $text_a$ written by a, and another $text_b$ written by user b. We may use the concatenation, input($attr_a$) + input($text_a$), to be the features of a positive example, and input($attr_a$) + input($text_b$) to be the features of a negative example. Note that the number of positive examples will always be the same as that of negative examples in our experiments. We also show performance of Node2Vec on the *AgreeWith* task for debate dataset by using the corresponding user network. We tuned the hyper-parameters p and q as described in [5].

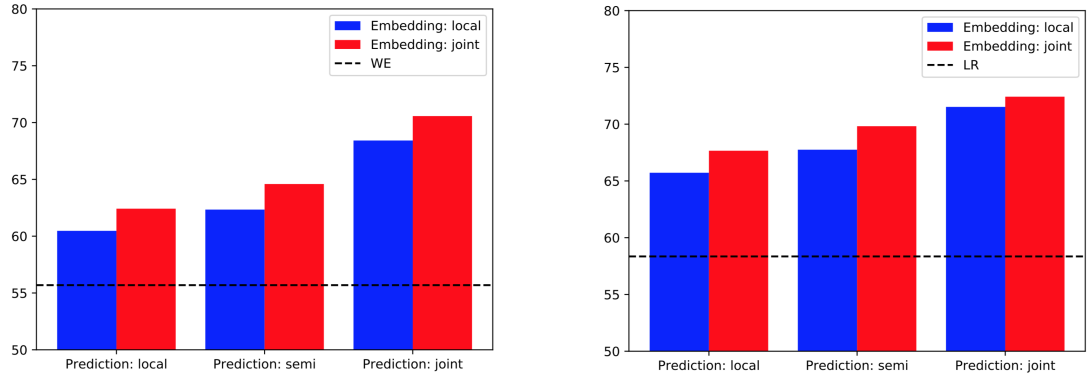
All other training methods consider different losses or a combination of them. The prediction method (second column) includes several variations. *Local* refers to considering

Table 5.4. Accuracy of prediction tasks under different settings on Facebook dataset

training method	prediction method	*prediction task* AuthoredBy	SameAuthor	TopFriend	EndorsedBy	average
WE-baseline	local	52.32	61.90	60.61	52.49	56.83 (55.30)
LR-baseline	local	51.94	62.30	60.21	58.96	58.35 (55.97)
SE-baseline	local	51.11	58.99	55.92	52.49	54.63 (53.65)
L_{TA}	local	66.00	65.94	69.70	61.23	65.72 (65.52)
L_{TT}		49.47	68.45	67.83	48.71	58.61 (55.30)
L_{AA}		49.61	61.15	70.82	48.37	57.49 (53.37)
$L(joint)$		65.24	68.42	75.29	61.31	67.66 (65.94)
L_{TA}	semi-joint	71.27	68.79	69.70	61.23	67.75 (69.33)
L_{TT}		49.85	71.39	67.83	48.71	59.44 (56.37)
L_{AA}		49.63	63.33	70.82	48.37	58.04 (54.00)
$L(joint)$		71.23	71.44	75.29	61.31	69.82 (70.11)
L_{TA}	joint	74.35	77.25	69.36	65.14	71.52 (73.93)
L_{TT}		49.19	77.12	68.62	47.92	60.71 (57.55)
L_{AA}		49.04	70.32	71.09	48.16	59.65 (55.64)
$L(joint)$		74.16	78.41	71.97	65.13	72.42 (74.07)

each prediction independently, while *semi-joint* (*joint*) enforces intra-task (and inter-task) constraints specified earlier. The final average column, reports the weighted average (by number of examples associated with each one of the prediction tasks) over the four prediction tasks.

The results reveal several interesting trends. First, we compare the impact of learning and using local vs. joint embedding. Local embedding, when used for predicting a task closely associated with the embedding objective, can help achieve a high accuracy. However, when we compared the averaged performance over *all* the prediction task, joint embedding always outperforms local embedding methods. These results are repeated in both the debate and facebook dataset, and when using either the local or joint prediction method. Second, joint vs. local prediction also reveals a similar trend. Joint prediction always outperforms local prediction. These results are repeated across all tasks, and in both datasets. The



(a) Debate. (b) Faceboook.

Figure 5.2. Comparison of best local vs. joint learning and prediction results on Debate (left) and Facebook (right) data.

best results are obtained when using the joint prediction method, over the joint embedding model.

Our results in Table 5.3 for debate dataset are summarized visually in Figure 5.2a. Specifically, it compares the average performance on the four tasks of the best locally trained model to the average performance of the jointly trained model, for the three different classification approaches (local, semi-joint and joint). Figure 5.2b summarizes the results for facebook dataset in Table 5.4 in the same manner. The black dotted line represents the best baseline performance (either using a supervised classifier over the same input representation, or an existing pre-trained embedding model). The plots clearly show that both joint learning of the embeddings and joint prediction using intra- and inter-task constraints consistently improve the overall accuracy of predictions. All of them outperform the best baseline model.

5.5.4 Variability Evaluation

In order to evaluate the variability of our model in terms of the number of training examples, we experimented with different amounts of training data. We presented prediction performance on *AuthoredBy* task for debate dataset and *SameAuthor* task for facebook dataset in Table 5.5. As we can see, the result continues to improve as more training data being used. This suggests that our model can benefit from more real world data. However, it

Table 5.5. Results of using different amount of training data on Debate and Facebook datasets

Dataset (prediction task)	training method	prediction method	20%	40%	60%
Debate (AuthoredBy)	LR-baseline	local	61.57	62.04	62.18
	L_{TA}	local	61.72	64.16	65.53
	$L(joint)$	joint	70.63	70.75	72.64
Facebook (SameAuthor)	LR-baseline	local	60.25	61.00	62.30
	L_{TT}	local	67.85	68.59	68.45
	$L(joint)$	joint	79.00	77.82	78.41

also worth noting that accuracy does not decrease dramatically when we reduce the training samples.

5.6 Discussion

In this work, we developed a model to embed users jointly in a latent representation, based on their social network information that includes attributes, interactions, and textual content. We considered both local and joint embeddings, as well as local and joint prediction methods that ensure predictions are consistent across the social network structure. We evaluate our approach on two real-world datasets, across four different prediction tasks. In contrast to previous work, where there has been a clear divide between the social and language analysis, our results indicate that using a holistic approach—combining users' information, their social behavior and language use into a joint model—produces consistent gains in predictive accuracy. We believe that studying natural language in its natural habitat, a social environment, can help shed light on many difficult tasks in which using text as the sole input results in a highly ambiguous tasks. These include challenging tasks such as discourse and conversational modeling, and other tasks which require pragmatic inferences.

6. MULTI-VIEW EDGE REPRESENTATION LEARNING IN SOCIAL NETWORKS

6.1 Introduction

Representation learning on large-scale information networks, such as social network data, has attracted a lot of attention recently because of its capabilities to convert a sparse and high-dimensional network structure into low-dimensional network embedding [1]. Learned node embeddings can be treated as features and used in various prediction and analysis tasks (e.g., node classification, link prediction and node visualization). Many node representation learning methods [2]–[5] have been proposed and demonstrated to be effective on preserving the structure of complex networks.

However, it is insufficient to use stand-alone node representations to represent the *edge* (i.e., relationship) between two nodes, especially when it needs to explain their relationship for various tasks. For this reason, edge representation learning—for describing the relationships between nodes—has recently gained increasing attention. However, in recent works, the edges between two nodes u and v is usually inferred coarsely and directly through a computation of an average, concatenation, or Hadamard product [21] of two node embeddings (e.g. $x_u \oplus x_v$ or $x_v \oplus x_u$). In this case, the same node embedding is used for every edge the node participates in (e.g., it is positioned ahead or behind the other according to the edge direction). Moreover, node relationships are often captured in a rough way, such as using distance or similarity measure. For example, the methods are unable to differentiate between $dist(x_u, x_v)$ and $dist(x_v, x_u)$ since they are equivalent.

To the best of our knowledge, there is a gap in network representation learning for social network data due to the lack of well-defined *edge* representations. Specifically, there are several drawbacks of heuristic edge approaches that consist of a combination of node representations. First, the node-centric approaches are unable to capture asymmetric relationships because they treat an edge $e_{u->v}$ the same as its reverse $e_{v->u}$. Since the characteristics of a node may vary according to its role, a node should be represented differently based on whether it is the source or the destination of an edge [6] rather than using a static representation.

Second, the node-centric approaches fail to preserve multiple types of proximities between two nodes in real world datasets. This does not account for different meanings of an edge between two nodes in different network *views* formed by user attributes (e.g., education, political stance). For example, two users could be connected in the *education* view because they went to the same school but disconnected in *occupation* view because they have different jobs. Previous node-centric methods fail to capture the nature of multiple views in social networks, and thus may be limited in their application to real-world data.

Third, the previous methods do not consider *interactions* between nodes, such as exchanged messages. Conversational factors (e.g., topical similarity, conversation frequency) has been shown to impact the quality of the node representation in social networks since it reflects *relational context* [100]. Therefore, node-centric approaches, which focus on learning a single node representation for all contexts, will be unable to describe relationships between users in varying contexts.

To address this, we introduce a comprehensive definition of edge representation and propose *MERL*, a novel multi-view low-rank edge representation with three main contributions: 1) MERL learns edge representations to reflect asymmetry, which are responsive to the difference between the source and destination roles of nodes; 2) MERL incorporates textual communications to identify the social signals in relationships (e.g. strength and affinity); and 3) MERL moderates the impact of different views between users that are observed in real-world *multi-view social networks*. Our experiments show MERL outperforms the state-of-the-art embedding models on link prediction and multilabel classification in social network data. We also analyze and compare the embeddings learned by MERL with previous methods. The results suggest that MERL embeddings are more correlated with the existence of edges in multi-view social networks than other models. In addition, our experiments show that MERL captures asymmetry of views and can distinguish the source/destination role of nodes better than other models.

6.2 Problem Definition

In this section, we formally define the problem and notation of multi-view edge representation learning.

Definition 6.2.1 ***Information Network.*** *An information network is defined as* $G = (V, E)$ *where* V *is a set of nodes and* E *is a set of edges between the nodes. Each edge* $\mathrm{e} \in E$ *is an ordered pair of two nodes, say* $\mathrm{e}_{uv} := (u, v)$ *where* $u, v \in V$ *and associated with a weight* $r_{uv} > 0$*, indicating the strength of relationship from* u *to* v*.*

When applying to *social network data* with attributes and textual interactions, the definition is extended to $G = (V, E, A, C)$ where V represents *users*, E represents *relationships* between users, A is a set of *user attributes* associated with each user and C is a collection of *exchanged messages* between the users. User attribute $a_u \in A$ is collected from user u's profile and group memberships. Document $c_{uv} \in C$ represents the set of posts t_{uv} sent from user u to user v.

Definition 6.2.2 ***Multi-View Information Network.*** *A* ***view*** *of a network is derived from a single type of relationship between the nodes, which can be characterized by a set of edges. We denote a* ***Multi-View Information Network*** *with* K *views as* $G = (V, E^1, E^2, ..., E^K)$*, where* $k \in \mathbb{Z}$ *and* $1 \leq k \leq K$*, is defined by user attributes* $a^k, \forall a^k \in A$*. Edge* e^k_{uv} *(associated with a weight* $r^k_{uv} > 0$*) exists if* $\mathrm{e}_{uv} \in G$ *and* $b^k(a^k_u, a^k_v) = 1$*, where* b^k *is a Boolean function test on whether an edge is formed in view* k *based on attribute values* a^k_u *and* a^k_v*. The functions defining edge formation vary for different views. In Figure 6.1, we use the function* b^h *in the Hometown view to test whether* u *and* v *have the same hometown, whereas the function* b^f *in the Friendship view tests whether* u *is in* v*'s top friend list.*

We also denote $C^k \subset C$, where $k \in K$, as a single (large) document of view k by collecting and concatenating exchanged posts c_{uv} and c_{vu} between user u and v, $\forall u, v \in V$ if they hold an edge in view k.

Definition 6.2.3 ***Node Embedding.*** *Given an information network* $G = (V, E)$*, the problem of node embedding aims to learn a low-dimensional vector representation* $x_u \in \mathbb{R}^d$ *for every node* $u \in V$ *where* $d \ll |V|$*, which preserves proximities between nodes.*

Now we consider the definition of edge representation. Some previous studies [47] straightforwardly learn an embedding vector for every edge in the network. Although this seems

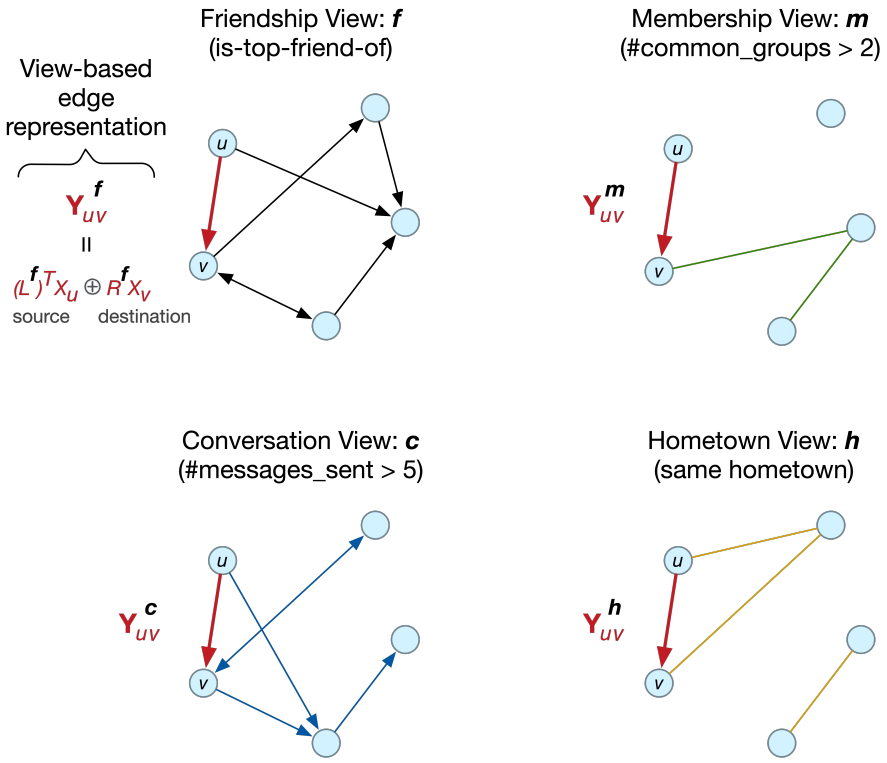


Figure 6.1. An example of edge representation in 4-view social network. User pairs have different affinity in different perspectives (views).

promising to capture the characteristics of edges, it is not scalable in social networks because it is computationally expensive to learn edge representations for all node pairs when $|V|$ is in the millions and requires an upper-bound space of $O(|V|^2)$. Furthermore, although learning a pair of edge embeddings respectively, say e_{uv} and e_{vu}, may describe the asymmetric relationship for a node pair u and v, it fails to explain the different characteristics in terms of u as edge source or u as edge destination in the network. To address the above shortcomings, we follow the work [6] to define an edge representation by a combination of its source and destination node with learning an asymmetric projection matrix for capturing the source-destination relationship between the nodes.

Definition 6.2.4 ***Edge Representation.*** *Given an information network $G = (V, E)$ and G's initial node embedding $x_u \in \mathbb{R}^d$, $\forall u \in V$. It aims to learn: (1) a low-rank asymmetric relational projection matrix $M = L \times R$, where $L \in \mathbb{R}^{d \times b}$, $R \in \mathbb{R}^{b \times d}$ and $b < d \ll |V|$; to produce*

(2) role-specific node representations $\tilde{x}_u^{src} = L^T x_u$ *and* $\tilde{x}_u^{dest} = Rx_u$*, where* $L^T x_u, Rx_u \in \mathbb{R}^b$*; and (3) low-dimensional edge representations* $\mathbf{Y}_{uv}$ *for each* (u, v)*, where* u *is source and* v *is destination node,* $\forall u, v \in V$*. The edge representation is defined as* $\mathbf{Y}_{uv} = \tilde{x}_u^{src} \oplus \tilde{x}_v^{dest}$ *and* $\oplus$ *is concatenation operator such that* $\mathbf{Y}_{uv} \in \mathbb{R}^{2b}$*. (i.e.,* $L^T x_u \oplus Rx_v$*).*

Note that this learns asymmetric transformations of the nodes, which generates two representations for a node u as part of edges: one when it is the source of a directed edge and one when it is a destination. This is completed through the projection matrix M which projects the node manifold coordinates to smaller space $\mathbb{R}^b$, since b is much smaller than d, we are able to further reduce the dimensionality. As described in previous section, another goal of this study is to leverage and preserve multiple types of proximities between two nodes. Therefore, we extend the definition of edge representation by adopting the concepts of multi-view networks [1] to multi-view edge representation.

Definition 6.2.5 ***Multi-View Edge Representation.*** *Given a multi-view information network* $G = (V, E^1, E^2, ..., E^K)$ *with* K *views, we define the multi-view edge representation for edge* (u, v) *as* $\mathbf{Y}_{uv}^{global} = \sum_{k=1}^{K} (\omega_{uv}^k \mathbf{Y}_{uv}^k)$*, where* $\mathbf{Y}_{uv}^k = (L^k)^T x_u \oplus R^k x_v$ *is the local edge representation of* (u, v) *and* ω_{uv}^k *is the weight of* (u, v) *for view* k*. We aim to jointly learn view-specific low-rank asymmetric relational projection matrices* $M^k = L^k \times R^k$*,* $\forall k \in K$*, where* $L^k \in \mathbb{R}^{d \times b}$*,* $R^k \in \mathbb{R}^{b \times d}$ *and* $b < d \ll |V|$*.*

For an edge (u, v), we define the local edge representation $\mathbf{Y}_{uv}^k$ in view k and the *single/global* edge representation $\mathbf{Y}_{uv}^{global}$ as a weighted linear combination of $\mathbf{Y}_{uv}^k$, $\forall k \in K$ across multiple views. The definition of weight parameter ω_{uv}^k varies in different tasks. In this work, we adopt the conversation factors introduced by [100] as the weight of node pairs in views. The details will be discussed in Section 6.3.1.

The multi-view nature is common in the graphs of social networks and contains rich information of the relationships among users. The different edge representations in multiple types of views are depicted in Figure 6.1. We show an example of social network with four views and the edges are created by different perspectives denoted in brackets. The two users u and v in the social network could form an edge (u, v) in the *Friendship* view because v sets u as a top friend, but they might not have an edge with regard to their different hometowns

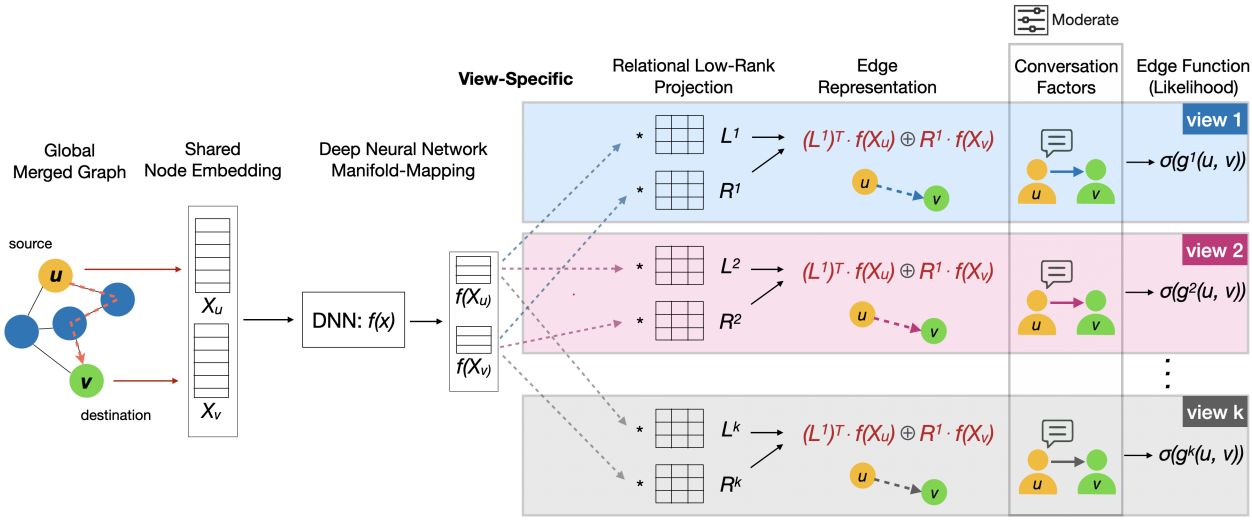


Figure 6.2. Overview of proposed MERL model.

in the *Hometown* view. In the meanwhile, u and v might chat often and have an edge in the *Conversation* view and join several common groups regarding the *Membership* view.

Definition 6.2.6 ***Global Merged Graph.*** *Given a multi-view information network* $G = (V, E^1, E^2, ..., E^K)$ *with* K *views, we define the global merged graph* $G_{global} = (V, E_{global})$ *by aggregating edges in* G*. We initialize the edge weight* r_{uv} *of edge* (u, v) *in* E_{global} *as:*

$$r_{uv} = \sum_{k=1}^{K} r_{uv}^{k} \tag{6.1}$$

where $u, v \in V$ and r_{uv}^{k} is edge weight of (u, v) in view k. The sparsity of each view is often quite different from others in multi-view networks. Therefore, learning node embeddings from a global merged graph enables a sparse view to consider complementary information from other dense views and increase the robustness of learned edge representations.

6.3 MERL: Multi-View Edge Representation Learning

We introduce *MERL*, a multi-view edge representation learning approach. The steps are summarized in Figure 6.2:

1. Jointly learn shared low-dimensional node embedding $x_v \in \mathbb{R}^D$, where $D \ll |V|$, for every node $v \in V$ in the global merged graph G_{global} aggregated from multiple-view graphs $G^k = (V, E^k), \forall k \in K$.

2. Take shared node embeddings x_v and feed them as input to a shared Deep Neural Network (DNN) $f_\theta : \mathbb{R}^D \to \mathbb{R}^d$ where $d < D$ for further mapping to lower-dimensional latent space. Based on [6], which found that learning node embeddings with DNN achieves better performance on link prediction than without DNN, we use a DNN to convert the dimension of the node embeddings from D to d rather than learning d-dimensional embeddings directly.

3. Jointly learn view-specific low-rank asymmetric relational projection matrices $M^k = L^k \times R^k$, where $L^k \in \mathbb{R}^{d \times b}$, $R^k \in \mathbb{R}^{b \times d}$ s.t. $b < d < D \ll |V|$ for K views and update shared node embedding x_v during learning process. The proximities of edges (u, v) and (v, u) across multiple views between a node pair u and v are preserved collectively.

4. Leverage the interaction factors (i.e., conversation similarity and conversation frequency) into the joint objective functions of edge representation learning to moderate the impacts of relations in different views.

6.3.1 Model Description

In the sections below, we describe how MERL introduces relational projection matrices to reflect view-specific asymmetry, leverages conversation factors to capture the interaction between nodes, and improves through initialization with pre-trained node embeddings.

Relational Projection Matrices

In order to capture the property of asymmetry in each view, we aim to learn multiple low-rank relational projection matrices. For the k-th view, we define the asymmetric relational projection matrix M^k, which represents the specific relation in view k by casting nodes to different view-specific domains. Meanwhile, we learn each edge $(u, v)^k$ in each view k by an edge function $g^k(u, v \,|\, f_\theta, \mathbf{x}, M^k) \in \mathbb{R}$, which comprises a multiplication between M^k and two

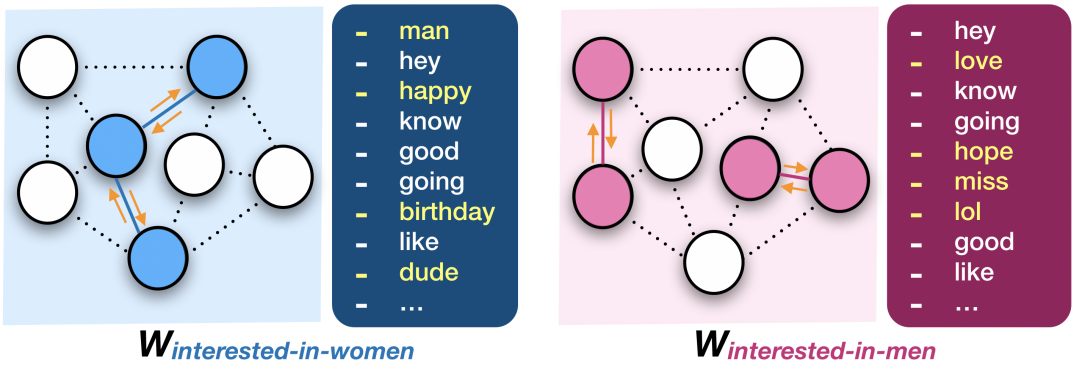


Figure 6.3. Word dictionaries of view *interested-in-women* and *interested-in-men* on Facebook dataset.

shared node embedding $f_\theta(x_u)$, $f_\theta(x_v)$, where $\theta = \{W_1, b_1, W_2, b_2, ...\}$ from a shared DNN. The low-rank affine projection matrix in each view k is defined as follows:

$$\begin{aligned} g^k(u, v \mid f_\theta, \mathbf{x}, M^k) &= \tilde{x}_u^{src^{(k)}} \cdot \tilde{x}_v^{dest^{(k)}} \\ &= (L^k)^T f_\theta(x_u) \cdot R^k f_\theta(x_v) \\ &= f_\theta(x_u)^T \times M^k \times f_\theta(x_v), \\ &\qquad \text{where } M^k = L^k \times R^k. \end{aligned} \tag{6.2}$$

Learning left (source) and right (destination) relational projections L^k and R^k not only enables us to represent a node differently as edge source or edge destination in view k but also help reduce dimensionality to smaller space. For example, the projection embedding of u is $(L^k)^T f_\theta(x_u) \in \mathbb{R}^b$ as edge source and $R^k f_\theta(x_u) \in \mathbb{R}^b$ as edge destination in view k, where $b < d$.

Conversation Factors

Since textual communication is the most common interaction between users in social networks, we adapt the conversation factors introduced by [100] to moderate the impact of different views for every user pair. First, we build a word dictionary $\mathbf{W_k}$ for each view $k \in K$ by using document C^k. We identify the most representative set of words with top-N TF-IDF [101] values in each document C^k and use these words as dictionary $\mathbf{W_k}$ for view k. Figure 6.3 shows the dictionaries of *interested-in-women* view and *interested-in-men* view

for the Facebook dataset. The words in yellow indicate the difference between two views, e.g. *interested-in-men* has more emotional words than *interested-in-women*, such as "love", "hope" and "miss".

Next, for each ordered user pair (u, v) that exists in view k, we capture the textual similarity of the messages sent from u to v by the Conversation Similarity Factor (μ^k_{uv}) and preserve their communication intensity by the Conversation Frequency Factor (ϕ^k_{uv}). The conversation similarity μ^k_{uv} is defined in Equation 6.3.

$$\mu^k_{u,v} = SIM(wv_{k_{uv}}, wv_{k_{vu}}) \tag{6.3}$$

where $wv_{k_{uv}}$ is the word existence vector that u sent to v in dictionary $\mathbf{W_k}$ of view k. For the SIM function, we follow [100] and use cosine as the similarity function. The similarity factor suggests whether the conversation between u and v is relevant to view k and reflects their degree of affinity in that view. Now we describe the conversation frequency ϕ^k_{uv} along with the $out^k(u, v)$ factor in Equation 6.4 and 6.5.

$$out^k(u, v) = \sum_{p=1}^{m} \frac{|w^k_p|}{|w_p|} \tag{6.4}$$

$$\phi^k_{uv} = \frac{out^k(u, v)}{\sum_{s=1}^{|V|} out^k(u, s)} \tag{6.5}$$

The $out^k(u, v)$ factor is the sum of fraction of words in $\mathbf{W_k}$ used in the messages from u to v. w_p is the set of words used in message p. w^k_p is the intersection of w_p and $\mathbf{W_k}$. Higher $out^k(u, v)$ means the words used in the textual communication from u to v are more relevant to view k. The frequency factor ϕ^k_{uv} indicates (u, v)'s intensity of interaction in view k. If u interacts more frequently with v compared to others, the frequency factor will be higher.

Node Initialization

As depicted in Figure 6.2, the initialization of shared node embedding could be random or pre-defined. Since the experiment results of [20] suggested that Node2Vec is more efficient than DeepWalk as the node initialization of their edge embedding model, we adopt Node2Vec as the pre-defined node initialization method and investigate if it's beneficial to MERL. Table 6.4 shows that the MERL variants with Node2Vec initialization outperform other

variants without Node2Vec initialization. Therefore, we conduct Node2Vec as the default node initialization of MERL in this work.

6.3.2 Model Optimization

In the training phase, we need sample global and view-specific positive and negative node pairs (edges) as the input to our multi-view model. First, we describe the training/testing data generation and negative sampling procedure.

Sampling from all weakly connected components

The training data in previous work [6] were sampled only from the largest weakly connected component (WCC) in graph, but there are many large components in our social network data. In fact, edges of each view are often sparse and form a connected component isolated from other views. We argue that sampling training data from only one largest connected component is not applicable to real world social networks since there are too many nodes and edges in other components will be removed and never be trained or tested. Therefore, we sample edges from all components when constructing our training and testing sets. Moreover, we extract random walks from all components instead of only from the largest connected component when sampling positive samples.

Sampling from multiple views

In order to adapt to multi-view data, we need to sample positive and negative samples for each view respectively rather than sampling a global set from G_{global}. It is because an edge (u, v) could exist in view A but does not show in view B. An edge could be a positive sample for view A but a negative one for view B. Thus, each edge (u, v) should be treated differently regarding different views.

Training/Testing data generation

First, given a global merged graph $G_{global} = (V, E_{global})$, we partition edges E_{global} into two equal-sized disjoint edge sets E_{train} and E_{test}. Both sets are sampled from all weakly

connected components. Second, negative sample set E^-_{train} is sampled from the compliment of E_{train} and E^-_{test} is sampled from the compliment of E_{global}. For directed graphs, we extend E^-_{test} to contain all edges (v, u) s.t. $(u, v) \in E_{global}$ and $(v, u) \notin E_{global}$. Third, we sample view-based positive sample sets E^k_{train} and E^k_{test} from E_{train} and E_{test} according to the edge information in the graph $G^k = (V, E^k)$ for each view k. The view-based negative sample sets E^{-k}_{train} and E^{-k}_{test} are sampled from E^-_{train} and E^-_{test}. Also, those edges that only exist in E_{train} and E_{test} but not in E^k_{train} and E^k_{test} are added to the negative sample sets. All the train/test edge sets in individual view k are of equal size.

The overall objective of *MERL* is aggregated by view-based objectives O_k shown as Equation 6.6, where $k \in K$, $\sigma(x) = 1/(1 + exp(x))$ is the standard logistic function, and the view-based edge function $g^k(u, v)$ is used to model edge existence as we introduce in Section 6.3.1. The first term maximizes the likelihood of existence for positive samples (u, v) and the second term minimizes the likelihood of N negative samples (u, v^-_{i}), uniformly sampled from E^{-k}_{train}.

$$\begin{aligned} O_k = -\sum_{(u,v)\in E^k} \Big[& \log \sigma(g^k(u, v)) \\ + \sum_{\substack{\mathrm{i}=1,\\ (u,v^-_{\mathrm{i}})\in E^{-k}_{train}}}^{N} & \log(1 - \sigma(g^k(u, v^-_{\mathrm{i}}))) \Big] \end{aligned} \tag{6.6}$$

After computing conversation factors (Section 6.3.1) for every user pair (u, v) in each view k, we use a tunable parameter α to assign different learning weights to the similarity factor μ^k_{uv} and frequency factor ϕ^k_{uv} in the form of $[1 + \alpha\mu^k_{uv} + (1 - \alpha)\phi^k_{uv}]$. The two factors play important roles to capture affinity and strength of (u, v)'s textual interactions and moderate the impacts of multiple views. We adopt PercentDelta [102] in mini-batch manner to minimize the overall objective function *jointly* over K views as shown in Equation 6.7:

$$\begin{aligned} O = \sum_{k=1}^{K} \Big\{ & -\sum_{(u,v)\in E^k} r^k_{uv}[1 + \alpha\mu^k_{uv} + (1 - \alpha)\phi^k_{uv}] \cdot \\ & \Big[\log \sigma(g^k(u, v)) + \sum_{\substack{\mathrm{i}=1,\\ (u,v^-_{\mathrm{i}})\in E^{-k}_{train}}}^{N} \log(1 - \sigma(g^k(u, v^-_{\mathrm{i}}))) \Big] \Big\} \end{aligned} \tag{6.7}$$

where r_{uv}^k is the weight of edge (u, v) in view k.

6.4 Experiments

We evaluate our proposed method *MERL* with several tasks including: 1) link prediction on *local (view-based) edge function* and 2) multilabel classification on *global edge representation*. In addition, we investigate how asymmetric relational projection, textual interactions (e.g. conversation factors) and the multi-view nature of MERL bring advantages to multi-view edge embedding learning.

6.4.1 Data

We analyze two social network data in our experiments:

- **Facebook:** The public Purdue Facebook network data was collected from March 2007 to March 2008, which includes 3 million post activities. For every edge (u, v) ever exists in at least one view, u posts at least one message (conversation) on v's timeline and vice versa. To evaluate the stability and sensibility regarding to the size of views, we use Facebook data to create two datasets for experiments. One multi-view network dataset with 6 views (denoted as *Facebook-6v*) and the other one with 41 views (denoted as *Facebook-41v*). The views are constructed from user profiles, group memberships and top friends information.

- **Twitter:** The Twitter dataset is sampled from the dataset collected by [89]. It contains 20 million post activities from June to July 2009. We use the posts with user mentions (e.g., "@david happy birthday!") as textual interactions. The 32 views are constructed from user profiles and follower/following information.

Table 6.1 shows statistics of three datasets. Edges in constructed views are weighted and directed, for example, if two users u and v share an attribute (e.g. *hometown-california*), two directed edges: (u, v) and (v, u) are created in the attribute-related view. Another example, if user u sets user j as top friend, there is a directed edge (j, u) in *is-top-friend-of* view.

Table 6.1. Statistics of multi-view datasets

Dataset	Facebook-6v	Facebook-41v	Twitter-32v
#Views	6	41	32
#Nodes	12,886	17,018	22,729
#Edges (top-2)	23,776 (interested-in-women)	29,818 (gender-male)	38,332 (unverified-account)
	8,556 (interested-in-men)	24,522 (looking-for-friendship)	36,883 (is-followed-by)
#Edges (bottom-2)	2,714 (political-view-democrat)	34 (hometown-california)	104 (language-german)
	1,376 (is-top-friend-of)	10 (relationship-status-complicated)	100 (hometown-australia)

Due to space limit, we list the views with top-2 most and bottom-2 least frequent edges respectively.

6.4.2 Experiment Settings

We compare our method with two types of related methods: single-view based and multi-view based.

- **MERL**: Our proposed *multi-view* edge representation learning method. We use the score of edge function $g^k(u, v)$ to predict existence of edge (u, v) in view k.
- **HeteroEdge** [20]: A *multi-view* edge representation learning method. Edge embedding in each view is generated by concatenating pre-trained node embedding. Aggregate all edge embeddings as features and feed into Neural Network to obtain the final edge embedding. Here we follow its best setting: Node2Vec-NeuralNet.
- **MVE** [1]: A *multi-view* node representation learning method. The final node embedding is derived from linear combination of embeddings in all views.

– **Asymmetric** [6]: A *single-view* representation learning method that introduces a low-rank asymmetric projection to transform nodes to source and destination representations. We perform the method and embed on the global merged graph G_{global}.

– **Node2Vec** [5]: A scalable *single-view* node representation learning method that maximizes the likelihood of preserving network neighborhoods of nodes. We perform it on merged graph G_{global}.

– **LINE** [3]: Another scalable *single-view* node representation learning method that preserves first- and second- order proximities of network structure. We report the results on merged graph G_{global}.

In order to have fair comparison, we train same dimension for all compared methods in each task. In training MERL, we use grid search to select optimal parameter setting. We optimize MERL through PercentDelta [102] for all trainable parameters with learning rate $= 0.5$ and $L2$ regularization $= 0.005$. For conversation factors, we set top-N TF-IDF $= 2000$ and $\alpha = 0.5$. Above settings are applied to all the datasets.

6.4.3 Link Prediction

Link prediction is a problem of inferring missing edges in a graph. We would like to evaluate the view-based edge score $g^k(u, v)$ served as the likelihood of the existence of edge (u, v) in view k. We follow and extend the setup in [5], [6] to conduct the experiments of link prediction. We compare MERL with two multi-view approaches (i.e. MVE, HeteroEdge) and three single-view approaches (i.e., Asymmetric, Node2Vec, LINE) with same dimension $d = 32$. Since both Facebook and Twitter datasets used in experiments are constructed in multi-view nature, the single-view approaches cannot be directly applied to the datasets. To tackle this problem, we perform single-view based methods on the weighted global merged graph G_{global}. That is, if a node pair has edges in three views, its edge weight is 3 in the global merged graph. The multi-view approaches learn from the multiple views in the original datasets, while the single-view approaches learn from the global merged graphs. Weighted ROC-AUC metrics are reported in Table 6.2. The results in bold statistically significant

outperform other methods ($p < 0.05$) in t-tests. We show MERL's results are promising and consistently outperform all other related works on all three datasets.

Table 6.2. The evaluation metrics of link prediction on datasets. Weighted ROC-AUC results are reported based on the ranking of (E^k_{test} , E^{-k}_{test}) across multiple views

Method	Facebook-6v	Facebook-41v	Twitter-32v
MERL	**0.894**	**0.838**	**0.841**
HeteroEdge	0.800	0.797	0.583
MVE	0.785	0.677	0.563
Asymmetric	0.658	0.595	0.652
Node2Vec	0.788	0.729	0.598
LINE	0.708	0.732	0.641

Table 6.3. Evaluation results of different settings on MERL model

Method	Node embedding initialization	Conversation factors	Facebook-6v	Facebook-41v	Twitter-32v
MERL	✓	✓	**0.894**	**0.838**	**0.841**
MERL-NoConv	✓		0.879	0.816	0.819
MERL-NoInit		✓	0.815	0.816	0.722
MERL-NoConvInit			0.803	0.796	0.708

Next, in order to further understand the extent of contribution of mechanisms used in MERL, we investigate how different settings impact the performance of MERL. The first mechanism we are interested in is node embedding initialization. We followed [20] to initialize MERL's node embeddings by Node2Vec. The second mechanism we want to evaluate is the conversation factors introduced by [100], which were described in previous sections. The ROC-AUC results are reported in Table 6.3. According to the results, we first surprisingly found that even without the two mechanisms (refer to "MERL-NoConvInit"), MERL still outperforms all other approaches. Second, Node2Vec initialization seems to play a more

important role on the performance enhancement than the conversation factors, especially for the Twitter dataset. The best MERL model is to combine Node2Vec initialization and use the conversation factors to moderate the impacts across multiple views.

6.4.4 Edge Embedding

We would like to evaluate the *global* edge representation $\mathbf{Y}_{uv}^{global}$, which is a *single* embedding for edge (u, v) where $u, v \in V$, learned by MERL. We perform *multilabel classification* to jointly predict edge existence in every single view via one-vs-rest logistic regression classifier and report the weighted average performance by taking number of edges in each view into consideration. For MERL, the dimension of edge embedding vector $d = 64$ and we take it as the input features to the classifier. For HeteroEdge, with Node2Vec-NeuralNet setting, we use $d = 16$ for pre-trained Node2Vec and output $d = 64$ from NeuralNet. For MVE, we use $d = 32$ during training and concatenate embeddings of source and destination nodes as the edge embedding resulted in $d = 64$ which is identical to our method. For Asymmetric method, it is performed on global merged graph G_{global} and we concatenate the learned representation with $d = 64$. For both Node2Vec and LINE, we perform Node2Vec and LINE on global merged graph G_{global} with $d = 32$ and also concatenate the source and destination nodes to form edge embedding with $d = 64$.

Table 6.4. The evaluation results of multilabel classification on datasets. Edge embedding is constructed and used as features in one-vs-rest logistic regression classifier

Method	Facebook-6v		Facebook-41v		Twitter-32v	
	Subset accuracy	Hamming score	Subset accuracy	Hamming score	Subset accuracy	Hamming score
MERL	**0.556**	**0.582**	**0.466**	**0.492**	**0.501**	**0.517**
HeteroEdge	0.546	0.573	0.377	0.458	0.315	0.395
MVE	0.489	0.507	0.445	0.454	0.416	0.445
Asymmetric	0.418	0.424	0.445	0.445	0.387	0.438
Node2Vec	0.521	0.543	0.447	0.462	0.367	0.462
LINE	0.477	0.497	0.444	0.458	0.395	0.499

The evaluation results are shown in Table 6.4. Two metrics are reported: *Subset accuracy* and *Hamming score* [91]. To evaluate an edge (u, v), let T be the true set of labels (e.g.,

$[1, 0, 0, 1]$ denotes the edge exists in the first and fourth views), S be the predicted set of labels. Subset accuracy get scores when S exactly match the corresponding set of labels in T, and Hamming score measure how close S is to T where $Hamming\_score(u, v) = |T \cap S|/|T \cup S|$. Overall, our experiments show that MERL outperforms other methods consistently in Subset accuracy and Hamming score metrics. The performance of HeteroEdge is comparable to our method on *Facebook-6v* but it drops on *Facebook-41v*, which indicates our method is more robust when the size of views is large. On *Twitter-32v*, MERL consistently outperforms other models while others struggle to achieve competitive subset accuracy scores.

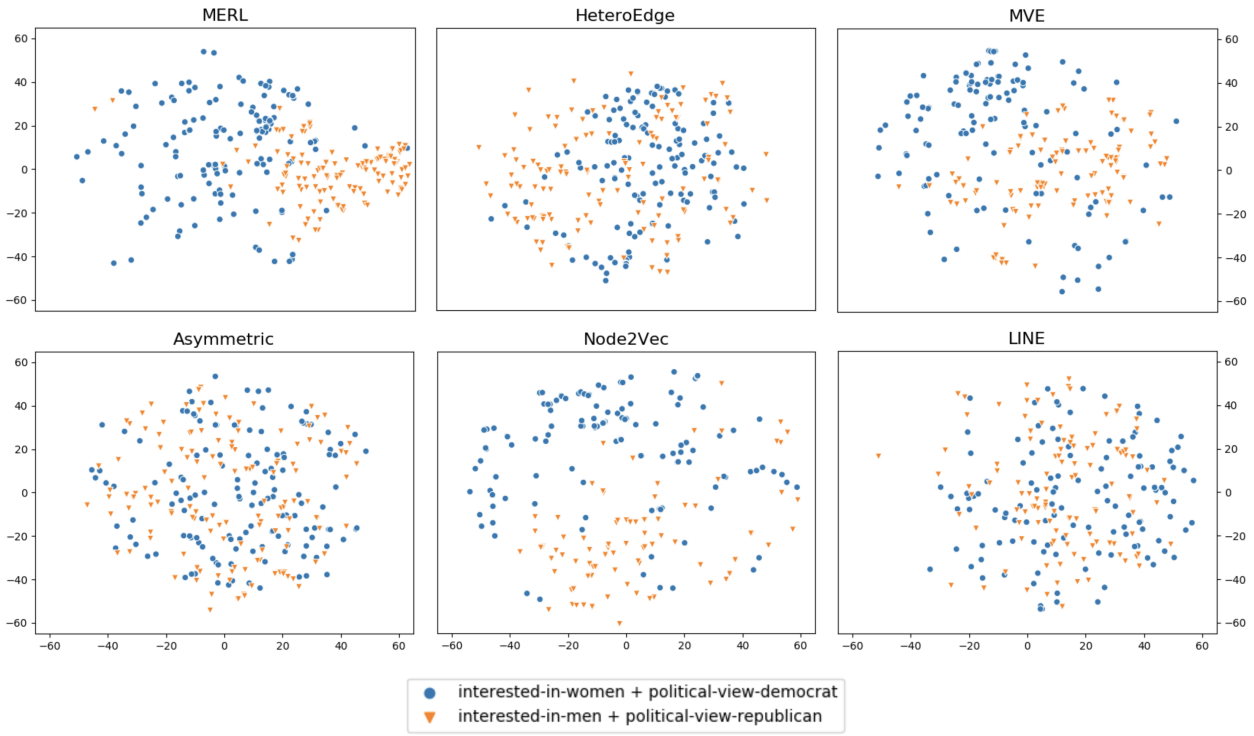


Figure 6.4. Edge embedding visualization on *Facebook-6v* dataset.

Following, we visualize the edge embeddings as shown in Figure 6.4. We are interested to know whether two edges (user pairs) hold similar relationship would have similar edge embedding, in other words, whether two edges would be embedded closer in the latent space. Here we only show the visualization results on *Facebook-6v* but the results of other datasets are also consistent. Samples are mapped to the reduced 2D space by using t-SNE [103] algorithm. The blue circles in the plots are user pairs who hold edges in two views *interested-in-women* and *political-view-democrat*, and orange triangles are user pairs

Table 6.5. Multi-view approach compared to original single-view method on *Facebook-6v* dataset

View	Graph Property				Multi-View	Single-View	
	#Node	#Edge	Graph Density	Largest Component	MERL-NoConvInit	Asymmetric	Asymmetric-all
is-senior-of	4667	5463	0.03%	3741	**0.6405**	0.4789	0.5337
is-top-friend-of	2113	1376	0.03%	32	**0.5906**	0.3472	0.5303
interested-in-women	7207	23776	0.05%	6485	**0.8302**	0.7654	0.7550
interested-in-men	3362	8556	0.08%	2824	**0.8486**	0.7749	0.7875
political-view-republican	1804	4208	0.13%	1274	**0.8184**	0.7891	0.7735
political-view-democrat	1385	2714	0.14%	844	0.7748	**0.8231**	0.7428

with edges in view *interested-in-men* and *political-view-republican*. Compared with other methods, MERL embeds same type of relationship closer and different type of relationships with distance. It is more effective to distinguish the two kinds of relationships. We also try other sets of examples and the visualization results are also similar.

6.4.5 Multi-View vs. Single-View

In the section, we investigate the effectiveness of multi-view embeddings by comparing MERL with *Asymmetric* [6]. To achieve a fair comparison, we use MERL-NoConvInit (removing both node embedding initialization and conversation factors) as our method. We analyze six views with different density levels in *Facebook-6v* dataset. The original *Asymmetric* proposed by [6] samples nodes only from the largest weakly connected component (WCC) in graph. We argue that it cannot adapt well to our datasets because the largest component of a view could be small and too many nodes are excluded. Taking the view *is-top-friend-of* in Table 6.5 as an example, its largest component contains only 32 out of 2113 nodes. In order to have further investigation, we implement *Asymmetric-all* and consider all weakly connected components in graph. We first learn multi-view embeddings by using MERL. Then we learn single-view embeddings for each of the six views by using *Asymmetric* and *Asymmetric-all* respectively. In other words, both Asymmetric- models are performed on view-based local graph G^k for each view k. ROC-AUC metrics are reported in Table 6.5. We have the following observations based on the results:

1. *Asymmetric-all* performs more consistently than *Asymmetric*, especially in views with low graph density (i.e., *is-senior-of*, *is-top-friend-of*). The results suggest that considering all components could alleviate the inconsistent performance of *Asymmetric* on sparse graphs. Although considering only the largest component could lead to better performance in dense graphs, the assumption of *Asymmetric* that ignoring nodes outside a graph's largest component could drop too many information and make the method inconsistent to fit real-world datasets.

2. MERL has a more consistent performance in all six views, and, interestingly, it outperforms *Asymmetric-all* on all six views and *Asymmetric* on all views except for *political-view-democrat*. Our results suggests that jointly learning embeddings from multiple views is more robust than learning from every single view respectively, especially when sparse views exist in the dataset. One possible reason is that the perspectives behind different views usually have correlation between each other (e.g., two users who share similar political views could be more likely to set each other as top friend), so MERL learns better embeddings for sparse views by leveraging information from dense views.

6.4.6 Asymmetric Relational Projection

We argue that many previous works directly used the concatenation of two node embeddings as their edge embeddings could not effectively describe the characteristics of edges since same embedding is used for source and destination role, especially in asymmetric views. For example, in the *is-top-friend-of* view, an user u could be another user v's top friend but u could have (or not) set v as his/her top friend. If we simply concatenate the node embeddings of u and v as their edge embeddings, say, $\mathrm{e}_{uv} = x_u \oplus x_v$ and $\mathrm{e}_{vu} = x_v \oplus x_u$, it is difficult to distinguish the difference between e_{uv} and e_{vu}. However, e_{uv} and e_{vu} should represent opposite characteristics. MERL tackles this ambiguity by asymmetric relational projection, which represents a node by different projections according to its role (i.e. source or destination). To examine this property, we conduct an experiment to compare MERL with Node2Vec and LINE, where the two compared methods are performed on view-based

local graph G^k for view k. We use *Facebook-6v* dataset and focus on the performance of asymmetric views *is-senior-of* and *is-top-friend-of*.

We construct view-based local edge embedding for each asymmetric view and take it as feature vector to logistic regression classifier, in order to predict the edge existence in the view. For MERL, we train on the whole *Facebook-6v* dataset and use learned local edge embedding Y^k_{uv} as input feature vector for edge (u, v). For Node2Vec and LINE, we concatenate the learned embeddings of source and destination node as input feature vector. The testing accuracy results are reported in Figure 6.5. We found that edge embeddings built by Node2Vec and LINE failed to predict the existence of edges in the two asymmetric views since their accuracy scores are all less or equal than 0.54. On the other hand, MERL achieves 0.82 in testing accuracy for *is-senior-of* view and 0.65 for *is-top-friend-of* view, which shows the effectiveness of MERL's view-based local edge embeddings on distinguishing edge existence in asymmetric views. Our results suggest that introducing asymmetric relational projection enables MERL adapt to the nature of asymmetric views better than other approaches without this property.

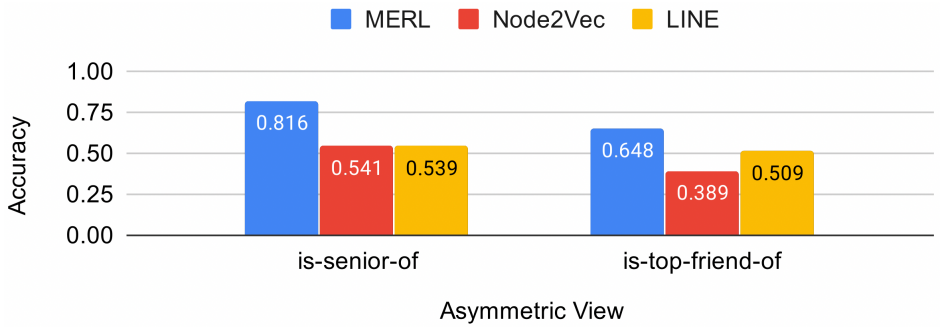


Figure 6.5. Edge existence prediction on asymmetric views on *Facebook-6v* dataset.

6.4.7 Conversation Factors

We have shown that conversation factors can effectively improve the performance of MERL in Table 6.3. Next we would like to evaluate the effectiveness of using conversation factors into MERL optimization process. We compare MERL and other models on the top-2 (i.e., *interested-in-women*, *interested-in-men*) and bottom-2 (i.e., *political-views-democrat*,

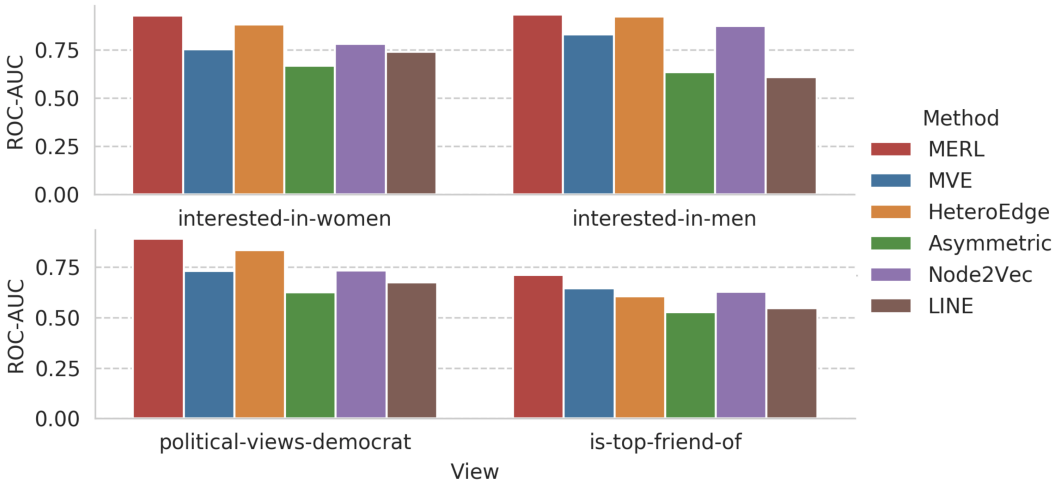


Figure 6.6. Evaluation results of link prediction on top-2 (top row) and bottom-2 (bottom row) views regarding the number of edges on *Facebook-6v* dataset.

is-to-friend-of) views with regard to the number of edges in view. The results are reported in Figure 6.6. We found that MERL consistently outperforms others in both dense and sparse views, while the performance of multi-view methods, such as HeteroEdge and MVE, are competitive in dense views but could be inconsistent in sparse views. Our results coincide with [100]'s findings that conversation factors enhance the robustness of embeddings for sparse views.

6.5 Related Work

6.5.1 Node Representation Learning

There has been increasing attention on low-dimensional graph embedding recently. Many approaches have been proposed to learn mode representation for data visualization, node classification, link prediction, and recommendation. DeepWalk [2] predicts the local neighborhood of nodes embeddings to learn graph embedding. LINE [3] learns feature representations in first- and second- order proximity respectively. GraRep [4] learns graph representation by optimizing k-step loss functions. Node2Vec [5] extends DeepWalk with a more sophisticated random walk procedure and explores diverse neighborhoods. Furthermore, several network embedding approaches have been proposed to preserve different network characteristics [104],

such as the structure and property [45], [105]–[109], side information [110]–[119], and other advanced information [120]–[123]. We compare our proposed method with Node2Vec and LINE.

6.5.2 Multi-View Node Representation Learning

Various methods proposed for learning representations with multiple views perform well on many applications such as clustering and recommendation. Previous works [56]–[59] are based on matrix factorization while there are expensive computational cost and not suitable for large-scale information networks. Clustering methods [60]–[64] neglect the collaboration and importance of different views in terms of weight learning. Deep learning methods [53], [65], [66] integrate different views via a linear/nonlinear combination. Attention-based approaches [1], [67] consider intra-view and/or inter-view importance to overcome above limitations. We compare and report results of MVE [1].

6.5.3 Edge Representation Learning

Previous work [47] keeps track of edges instead of nodes when performing random walks and learns edge representation using word2vec-like method for community detection. Another method [49] uses topological features for edge attributed clustering problem but is rather complex and thus not very interpretable. [6] models edges as functions of node embeddings and jointly learns the edge function and node embeddings by optimizing their model using graph likelihood. [50] characterizes edges in complex networks by signed graphs and vector-valued graphs. Two learning biases are introduced for directed and undirected signed graphs respectively. [51] proposed edge2vec, which represents graphs considering edge semantics. ENRNM [52] learns the formation mechanism of link by combining the representations of edge and nodes as the full representation of link. [20] built a friend recommendation system using edge embedding on social networks. Network representation is learned based on edge heterogeneity in multi-graphs. To the best of our understanding, we have identified several research gaps from previous studies: only [6] discussed a node should have different representations regarding its role in an edge, however, none of existing edge

representation learning models has considered the multi-view nature of social networks as well as the textual interactions between users. We compare results with [6] and [20].

6.6 Discussion

Network embedding models aim to learn low-dimensional representations for nodes and/or edges in graphs. For social networks, learning *edge representations* is especially beneficial as we need to describe or explain the relationships, activities, and interactions between users. Existing approaches that learn stand-alone node embeddings, and represent edges as *pairs* of node embeddings, are limited in their applicability because nodes participate in multiple relationships, which should be considered. Besides, social networks often contain multiple types of edges, which yields multi-view contexts that need to be considered in the representation. In this work, we propose a new methodology, *MERL*, that (1) captures *asymmetry* in *multiple views* by learning well-defined edge representations that are responsive to the difference between the source and destination node roles, and (2) incorporates *textual communications* to identify multiple sources of social signals (e.g. strength and affinity) that moderate the impact of different views between users. Our experiments show that MERL outperforms alternative state-of-the-art embedding models on link prediction (11.75%, 5.14% and 28.99% increase in ROC-AUC compared the best baseline respectively on three datasets) and multilabel classification tasks across multiple views in social network datasets. We further analyze the learned embeddings of MERL and demonstrate they are more correlated with the existence of view-based edges compared to previous methods.

7. CONCLUSION

While previous work in network representation learning has recently developed many promising methods, these works are still limited in their ability to incorporate textual communication and to approach edge representation learning in a manner that better captures the unique characteristics of relationships in social networks. This dissertation aims to address these limitations.

We first presented a preliminary study towards finding a joint node representation by capturing the contents of interactions—through computing the textual similarity between the messages generated by each one of the nodes. Our experimental results support the conjecture that new learned representations can help improve predictive performance in two social relations prediction tasks.

Second, we proposed a novel relationship embedding model, *TransConv*, which is built upon structural translation on relationship hyperplanes and further optimized through conversation factors established from textual communications. To the best of our knowledge, *TransConv* is the first model that considers both intensity and similarity of textual communications between users. Our experiments show that *TransConv* outperforms the state-of-the-art relationship embedding models in the tasks of social network completion, triplets classification and multilabel classification.

Third, we developed a model to embed users jointly in a latent representation, based on their social network information that includes attributes, interactions, and textual content. Our work explored the connections between text and user attributes, attempting to create a common representation for the two. We considered both local and joint embeddings, as well as local and joint prediction methods that ensure predictions are consistent across the social network structure.

Finally, we introduced a comprehensive definition of edge representation and propose *MERL*, a novel multi-view edge representation learning. *MERL* has three key characteristics, in that it: (i) learns edge representations to reflect asymmetry, which is responsive to the difference between the source and destination roles of nodes; (ii) incorporates textual communications to identify the social signals in relationships (e.g. strength and affinity); and

(iii) moderates the impact of different views between users that are observed in real-world multi-view social networks. Our experiments show *MERL* outperforms the state-of-the-art embedding models on link prediction and multilabel classification in social network data. We also analyze and compare the embeddings learned by *MERL* with previous methods. The results suggest that *MERL* embeddings are more correlated with the existence of edges in multi-view social networks than other models. In addition, our experiments show that *MERL* captures asymmetry of views and can distinguish the source/destination role of nodes better than other models.

7.1 Contribution

To summarize, the contributions of this dissertation include the following:

- Models and Frameworks
  - Development of *TransConv*—a network representation learning method that incorporates textual interactions between pairs of users to improve representation learning of both users and relationships.
  - Development of a joint network representation learning method to embed users jointly in a latent representation, based on their social network information that includes attributes, interactions, and textual content.
  - Development of *MERL*—a multi-view edge representation learning method that incorporates textual communications, asymmetry, and edge types to better capture the social signals in relationships.
- Algorithmic Contributions
  - Development of *conversational factors* to reflect the degree of affinity and intensity observed in textual interactions between pairs of social network users, used in *TransConv* and *MERL*.
  - Development of deep structured joint embedding method with multiple embedding views and objectives.

  - Development of view-specific relational projection matrices in *MERL*.

- Evaluation

  - Development of several evaluation criteria to explore the utility of learned network representations (e.g. node, edge and relationship embeddings).

7.2 Future Work

There are several directions to pursue after the completion of this dissertation. First, for multi-relational network representation learning in Chapter 4, we can explore the relations over heterogeneous networks with multiple node and edge types, and moreover, identify and distinguish higher-order relations. Homogeneous network representation approaches only consider a singular type of nodes and relations, yet a large number of social and information networks are heterogeneous in nature. In order to explain real-world network interactions better, it is worth exploring this topic further under heterogeneous contexts, which contain more complex relationships.

Second, for joint semantic embedding learning in Chapter 5, we can develop a generalized methodology for the trade-offs between different input representations. We can extend the idea of *node-wise* that is used here to a *pair-wise* perspective. That is, studying the connections between users' natural language communications with their attributes agreements. The promising results we observed in the *node-wise* viewpoint encourage us to seek deeper into this topic in the future. For example, understanding how common attributes that a pair of users hold influences and interacts with the textual conversations exchanged by them, and how they interact and correlate with other pairs with similar attribute agreements or textual conversations.

Third, for multi-view network representation learning in Chapter 6, we can investigate the correlation among multiple views, and leverage other mechanisms to identify the strength of social signals in different views. For example, an attention mechanism would allow a model to concentrate more on a few task-relevant factors and thus could help determine the signal intensity across multiple views. In addition, we can explore a wider range of conversational signals when incorporating textual communication into proposed network representation

learning models (e.g., leveraging temporal textual interactions). Our work has shown that the proposed conversation similarity and frequency factors can effectively improve predictive performance in our experiments. Therefore, investigating additional natural language approaches, to develop a wider range of conversational factors, is a promising direction to explore.

REFERENCES

[1] M. Qu, J. Tang, J. Shang, X. Ren, M. Zhang, and J. Han, "An attention-based collaboration framework for multi-view network representation learning," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, ACM, 2017, pp. 1767–1776.

[2] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2014, pp. 701–710.

[3] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*, International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[4] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, ACM, 2015, pp. 891–900.

[5] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2016, pp. 855–864.

[6] S. Abu-El-Haija, B. Perozzi, and R. Al-Rfou, "Learning edge representations via low-rank asymmetric projections," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, ACM, 2017, pp. 1787–1796.

[7] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, 2013, pp. 2787–2795.

[8] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes.," in *AAAI*, vol. 14, 2014, pp. 1112–1119.

[9] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion.," in *AAAI*, vol. 15, 2015, pp. 2181–2187.

[10] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, vol. 1, 2015, pp. 687–696.

[11] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[12] R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun, "Representation learning of knowledge graphs with entity descriptions.," in *AAAI*, 2016, pp. 2659–2665.

[13] H. Xiao, M. Huang, L. Meng, and X. Zhu, "Ssp: Semantic space projection for knowledge graph embedding with text descriptions.," in *AAAI*, vol. 17, 2017, pp. 3104–3110.

[14] D. K. Elson, N. Dames, and K. R. McKeown, "Extracting social networks from literary fiction," in *Proceedings of ACL*, Association for Computational Linguistics, 2010, pp. 138–147.

[15] M. Van De Camp and A. van den Bosch, "A link to the past: Constructing historical social networks," in *Proceedings of the Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, Association for Computational Linguistics, 2011, pp. 61–69.

[16] A. Agarwal, A. Corvalan, J. Jensen, and O. Rambow, "Social network analysis of alice in wonderland," in *Proceedings of the Workshop on Computational Linguistics for Literature*, 2012.

[17] R. West, H. S. Paskov, J. Leskovec, and C. Potts, "Exploiting social network structure for person-to-person sentiment analysis," *TACL*, 2014.

[18] Y. Yang and J. Eisenstein, "Putting things in context: Community-specific embedding projections for sentiment analysis," *arXiv preprint arXiv:1511.06052*, vol. 4, no. 3, 2015.

[19] H. Chen, X. Sun, Y. Tian, B. Perozzi, M. Chen, and S. Skiena, "Enhanced network embeddings via exploiting edge labels," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, ACM, 2018, pp. 1579–1582.

[20] J. Verma, S. Gupta, D. Mukherjee, and T. Chakraborty, "Heterogeneous edge embeddings for friend recommendation," in *European Conference on Information Retrieval*, Springer, 2019.

[21] R. A. Horn, "The hadamard product," in *Proc. Symp. Appl. Math*, vol. 40, 1990, pp. 87–169.

[22] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data.," in *ICML*, vol. 11, 2011, pp. 809–816.

[23] R. Jenatton, N. L. Roux, A. Bordes, and G. R. Obozinski, "A latent factor model for highly multi-relational data," in *Advances in Neural Information Processing Systems*, 2012, pp. 3167–3175.

[24] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "A semantic matching energy function for learning with multi-relational data," *Machine Learning*, vol. 94, no. 2, pp. 233–259, 2014.

[25] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *Advances in neural information processing systems*, 2013, pp. 926–934.

[26] J. Weston, A. Bordes, O. Yakhnenko, and N. Usunier, "Connecting language and knowledge bases with embedding models for relation extraction," *arXiv preprint arXiv:1307.7973*, 2013.

[27] S. Riedel, L. Yao, A. McCallum, and B. M. Marlin, "Relation extraction with matrix factorization and universal schemas," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013, pp. 74–84.

[28] M. Nickel, V. Tresp, and H.-P. Kriegel, "Factorizing yago: Scalable machine learning for linked data," in *Proceedings of the 21st international conference on World Wide Web*, ACM, 2012, pp. 271–280.

[29] Q. Wang, B. Wang, and L. Guo, "Knowledge base completion using embeddings and rules," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[30] Z. Wei, J. Zhao, K. Liu, Z. Qi, Z. Sun, and G. Tian, "Large-scale knowledge base completion: Inferring via grounding network sampling over selected instances," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, ACM, 2015, pp. 1331–1340.

[31] S. Guo, Q. Wang, B. Wang, L. Wang, and L. Guo, "Semantically smooth knowledge graph embedding," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, vol. 1, 2015, pp. 84–94.

[32] Y. Lin, Z. Liu, H. Luan, M. Sun, S. Rao, and S. Liu, "Modeling relation paths for representation learning of knowledge bases," *arXiv preprint arXiv:1506.00379*, 2015.

[33] K. Guu, J. Miller, and P. Liang, "Traversing knowledge graphs in vector space," *arXiv preprint arXiv:1506.01094*, 2015.

[34] K. Toutanova, V. Lin, W.-t. Yih, H. Poon, and C. Quirk, "Compositional learning of embeddings for relation paths in knowledge base and text," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2016, pp. 1434–1444.

[35] H. Zhong, J. Zhang, Z. Wang, H. Wan, and Z. Chen, "Aligning knowledge and text embeddings by entity descriptions," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 267–272.

[36] Z. Wang and J.-Z. Li, "Text-enhanced representation learning for knowledge graph.," in *IJCAI*, 2016, pp. 1293–1299.

[37] S. Guo, Q. Wang, L. Wang, B. Wang, and L. Guo, "Jointly embedding knowledge graphs and logical rules," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 192–202.

[38] T. Rocktäschel, S. Singh, and S. Riedel, "Injecting logical background knowledge into embeddings for relation extraction," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015, pp. 1119–1129.

[39] H. Chen, B. Perozzi, R. Al-Rfou, and S. Skiena, "A tutorial on network embeddings," *arXiv preprint arXiv:1808.02590*, 2018.

[40] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[41] T. F. Cox and M. A. Cox, *Multidimensional scaling*. Chapman and hall/CRC, 2000.

[42] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[43] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *arXiv preprint arXiv:1709.05584*, 2017.

[44] P. D. Hoff, A. E. Raftery, and M. S. Handcock, "Latent space approaches to social network analysis," *Journal of the american Statistical association*, vol. 97, no. 460, pp. 1090–1098, 2002.

[45] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2016, pp. 1225–1234.

[46] M. Zhang and Y. Chen, "Weisfeiler-lehman neural machine for link prediction," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2017, pp. 575–583.

[47] S. Li, H. Zhang, D. Wu, C. Zhang, and D. Yuan, "Edge representation learning for community detection in large scale information networks," in *International Workshop on Mobility Analytics for Spatio-temporal and Social Data*, Springer, 2017, pp. 54–72.

[48] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[49] N. K. Ahmed, R. A. Rossi, T. L. Willke, and R. Zhou, "Edge role discovery via higher-order structures," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2017, pp. 291–303.

[50] G. Le Falher, "Characterizing edges in signed and vector-valued graphs," Ph.D. dissertation, Université de Lille, 2018.

[51] Z. Gao, G. Fu, C. Ouyang, S. Tsutsui, X. Liu, and Y. Ding, "Edge2vec: Learning node representation using edge semantics," *arXiv preprint arXiv:1809.02269*, 2018.

[52] G. Xu, X. Wang, Y. Wang, D. Lin, X. Sun, and K. Fu, "Edge-nodes representation neural machine for link prediction," *Algorithms*, vol. 12, no. 1, p. 12, 2019.

[53] Y. Shi, F. Han, X. He, X. He, C. Yang, J. Luo, and J. Han, "Mvn2vec: Preservation and collaboration in multi-view network embedding," *arXiv preprint arXiv:1801.06597*, 2018.

[54] J. Li, A. Ritter, and D. Jurafsky, "Learning multi-faceted representations of individuals from heterogeneous evidence using neural networks," *CoRR*, 2015.

[55] A. Benton, R. Arora, and M. Dredze, "Learning multiview embeddings of twitter users," in *ACL*, 2016.

[56] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2008, pp. 650–658.

[57] D. Greene and P. Cunningham, "A matrix factorization approach for integrating multiple data views," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2009, pp. 423–438.

[58] J. Liu, C. Wang, J. Gao, and J. Han, "Multi-view clustering via joint nonnegative matrix factorization," in *Proceedings of the 2013 SIAM International Conference on Data Mining*, SIAM, 2013, pp. 252–260.

[59] D. Greene and P. Cunningham, "Producing a unified graph representation from multiple social network views," in *Proceedings of the 5th annual ACM web science conference*, ACM, 2013, pp. 118–121.

[60] D. Zhou and C. J. Burges, "Spectral clustering and transductive learning with multiple views," in *Proceedings of the 24th international conference on Machine learning*, ACM, 2007, pp. 1159–1166.

[61] K. Chaudhuri, S. M. Kakade, K. Livescu, and K. Sridharan, "Multi-view clustering via canonical correlation analysis," in *Proceedings of the 26th annual international conference on machine learning*, ACM, 2009, pp. 129–136.

[62] T. Xia, D. Tao, T. Mei, and Y. Zhang, "Multiview spectral embedding," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, no. 6, pp. 1438–1446, 2010.

[63] A. Kumar, P. Rai, and H. Daume, "Co-regularized multi-view spectral clustering," in *Advances in neural information processing systems*, 2011, pp. 1413–1421.

[64] G. Ma, L. He, C.-T. Lu, W. Shao, P. S. Yu, A. D. Leow, and A. B. Ragin, "Multi-view clustering with graph embedding for connectome analysis," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, ACM, 2017, pp. 127–136.

[65] W. Wang, R. Arora, K. Livescu, and J. Bilmes, "On deep multi-view representation learning," in *International Conference on Machine Learning*, 2015, pp. 1083–1092.

[66] J. Ni, S. Chang, X. Liu, W. Cheng, H. Chen, D. Xu, and X. Zhang, "Co-regularized deep multi-network embedding," in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW '18, 2018, pp. 469–478.

[67] Y. Wang, L. Hu, Y. Zhuang, and F. Wu, "Intra-view and inter-view attention for multi-view network embedding," in *Pacific Rim Conference on Multimedia*, Springer, 2018, pp. 201–211.

[68] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 689–696.

[69] G. Andrew, R. Arora, J. Bilmes, and K. Livescu, "Deep canonical correlation analysis," in *International conference on machine learning*, 2013, pp. 1247–1255.

[70] J. Leskovec, L. Backstrom, R. Kumar, and A. Tomkins, "Microscopic evolution of social networks," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2008, pp. 462–470.

[71] R. Kumar, J. Novak, and A. Tomkins, "Structure and evolution of online social networks," in *Link mining: models, algorithms, and applications*, Springer, 2010, pp. 337–357.

[72] R. Xiang, J. Neville, and M. Rogati, "Modeling relationship strength in online social networks," in *Proceedings of the 19th international conference on World wide web*, ACM, 2010, pp. 981–990.

[73] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in *Proceedings of the 19th international conference on World wide web*, ACM, 2010, pp. 641–650.

[74] C. Danescu-Niculescu-Mizil, L. Lee, B. Pang, and J. Kleinberg, "Echoes of power: Language effects and power differences in social interaction," in *Proceedings of WWW*, 2012, pp. 699–708.

[75] A. Hassan, A. Abu-Jbara, and D. Radev, "Detecting subgroups in online discussions by modeling positive and negative relations among participants," in *Proceedings of EMNLP-CoNLL*, Association for Computational Linguistics, 2012, pp. 59–70.

[76] K. Filippova, "User demographics and language in an implicit social network," in *Proceedings of EMNLP-CoNLL*, Association for Computational Linguistics, 2012, pp. 1478–1488.

[77] S. Volkova, G. Coppersmith, and B. Van Durme, "Inferring user political preferences from streaming communications.," in *Proceedings of ACL*, 2014, pp. 186–196.

[78] A. Rahimi, D. Vu, T. Cohn, and T. Baldwin, "Exploiting text and network context for geolocation of social media users," in *Proceedings of NAACL*, 2015.

[79] S. Volkova, Y. Bachrach, M. Armstrong, and V. Sharma, "Inferring latent user properties from texts published in social media.," in *Proceedings of AAAI*, 2015.

[80] V. Krishnan and J. Eisenstein, "" you're mr. lebowski, i'm the dude": Inducing address term formality in signed social networks," *arXiv preprint arXiv:1411.4351*, 2014.

[81] M. Sap, G. Park, J. C. Eichstaedt, M. L. Kern, D. Stillwell, M. Kosinski, L. H. Ungar, and H. A. Schwartz, "Developing age and gender predictive lexica over social media," 2014.

[82] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations.," in *Proceedings of NAACL*, 2013.

[83] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of EMNLP*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162.

[84] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, pp. 415–444, 2001.

[85] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[86] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[87] G. Salton and M. J. McGill, "Introduction to modern information retrieval," 1986.

[88] A. Garcia-Duran, R. Gonzalez, D. Onoro-Rubio, M. Niepert, and H. Li, "Transrev: Modeling reviews as translations from users to items," *arXiv preprint arXiv:1801.10095*, 2018.

[89] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?" In *Proceedings of the 19th international conference on World wide web*, AcM, 2010, pp. 591–600.

[90] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[91] S. Godbole and S. Sarawagi, "Discriminative methods for multi-labeled classification," in *Pacific-Asia conference on knowledge discovery and data mining*, Springer, 2004, pp. 22–30.

[92] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *WWW*, 2013.

[93] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3128–3137.

[94] L. Wang, Y. Li, and S. Lazebnik, "Learning deep structure-preserving image-text embeddings," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5005–5013.

[95] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, "Learning deep structured semantic models for web search using clickthrough data," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, ACM, 2013, pp. 2333–2338.

[96] E. Amid and A. Ukkonen, "Multiview triplet embedding: Learning attributes in multiple maps," in *International Conference on Machine Learning*, 2015, pp. 1472–1480.

[97] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, 2016.

[98] I. Gurobi Optimization, *Gurobi optimizer reference manual*, 2016. [Online]. Available: http://www.gurobi.com.

[99] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Advances in neural information processing systems*, 2015, pp. 3294–3302.

[100] Y.-Y. Lai, J. Neville, and D. Goldwasser, "Transconv: Relationship embedding in social networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4130–4138.

[101] G. Salton, "Developments in automatic text retrieval," *science*, vol. 253, no. 5023, pp. 974–980, 1991.

[102] S. Abu-El-Haija, "Proportionate gradient updates with percentdelta," *arXiv preprint arXiv:1708.07227*, 2017.

[103] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[104] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2018.

[105] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[106] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[107] S. Chen, S. Niu, L. Akoglu, J. Kovačević, and C. Faloutsos, "Fast, warped graph embedding: Unifying framework and one-click algorithm," *arXiv preprint arXiv:1702.05764*, 2017.

[108] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 1105–1114.

[109] S. Wang, J. Tang, C. Aggarwal, Y. Chang, and H. Liu, "Signed network embedding in social media," in *Proceedings of the 2017 SIAM international conference on data mining*, SIAM, 2017, pp. 327–335.

[110] M. Ou, P. Cui, F. Wang, J. Wang, and W. Zhu, "Non-transitive hashing with latent similarity components," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 895–904.

[111] C. Tu, W. Zhang, Z. Liu, M. Sun, *et al.*, "Max-margin deepwalk: Discriminative learning of network representation.," in *IJCAI*, vol. 2016, 2016, pp. 3889–3895.

[112] T. M. Le and H. W. Lauw, "Probabilistic latent document network embedding," in *2014 IEEE International Conference on Data Mining*, IEEE, 2014, pp. 270–279.

[113] X. Sun, J. Guo, X. Ding, and T. Liu, "A general framework for content-enhanced network representation learning," *arXiv preprint arXiv:1610.02906*, 2016.

[114] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-party deep network representation," *Network*, vol. 11, no. 9, p. 12, 2016.

[115] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 2017, pp. 731–739.

[116] Y. Jacob, L. Denoyer, and P. Gallinari, "Learning latent representations of nodes for classifying in heterogeneous social networks," in *Proceedings of the 7th ACM international conference on Web search and data mining*, 2014, pp. 373–382.

[117] Z. Huang and N. Mamoulis, "Heterogeneous information network embedding for meta path based proximity," *arXiv preprint arXiv:1701.05291*, 2017.

[118] L. Xu, X. Wei, J. Cao, and P. S. Yu, "Embedding of embedding (eoe) joint embedding for coupled heterogeneous networks," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 2017, pp. 741–749.

[119] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2015, pp. 119–128.

[120] C. Li, J. Ma, X. Guo, and Q. Mei, "Deepcas: An end-to-end predictor of information cascades," in *Proceedings of the 26th international conference on World Wide Web*, 2017, pp. 577–586.

[121] R. Hu, C. C. Aggarwal, S. Ma, and J. Huai, "An embedding approach to anomaly detection," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, IEEE, 2016, pp. 385–396.

[122] T. Man, H. Shen, S. Liu, X. Jin, and X. Cheng, "Predict anchor links across social networks via an embedding approach.," in *IJCAI*, vol. 16, 2016, pp. 1823–1829.

[123] S. Bourigault, C. Lagnier, S. Lamprier, L. Denoyer, and P. Gallinari, "Learning social network embeddings for predicting information diffusion," in *Proceedings of the 7th ACM international conference on Web search and data mining*, 2014, pp. 393–402.

VITA

Yi-Yu (Ellen) Lai was born and raised in Taipei, Taiwan. She received her Bachelor's and Master's degree from the Department of Information Management at National Central University and National Taiwan University, respectively. She later earned her Master's and Ph.D. degrees from the Department of Computer Science at Purdue University. Her research interests lie in the fields of data mining, machine learning, and natural language processing for information networks, focusing on relational representation learning. She has published several works at various conferences and journals. She also has years of industrial experience, worked as a machine learning researcher at Intel and a software engineer at Trend Micro.