Link Prediction in Knowledge Graphs

Joseph Cappadona (jc10762), Connor Reed (cr3221)

Inference and Representation, Fall 2021

Abstract

Knowledge graphs represent a special kind of directed graph in which entities (nodes) are connected by labeled relations (edges). An important and longstanding task in the domain of knowledge graphs is predicting new links between entities given an incomplete subset of the relations in the graph. In this paper, we survey an evolution of approaches that have been proposed and tested for link prediction which exploit existing relations and features of nodes within a knowledge graph, including relational Markov networks, path ranking algorithms, stochastic relational models, and message passing via graph neural networks.

Contents

1 Introduction

1.1 Knowledge graphs

A *knowledge graph* (KG) is a graph defined over a set of *entities* $\mathcal{E}$, corresponding to vertices, and a set of *relations* $\mathcal{R}$, corresponding to edges. A *relation triplet* or *fact* is a triplet of the form (h, r, t) where $h, t \in \mathcal{E}$ and $r \in \mathcal{R}$. We then define $\mathcal{G}_{\text{gold}} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ as the set of *gold* facts; that is, $\mathcal{G}_{\text{gold}}$ is the set of all relation triplets which hold for entities and relations. If a triplet is in $\mathcal{G}_{\text{gold}}$, it is a *positive triplet*, otherwise, it is a *negative triplet*.

KGs represent a useful abstraction for organizing information with many real-world applications. They are able to mine, organize, and manage knowledge from large-scale data to improve the quality of information services for users, including in tasks such as information retrieval, recommendation, question answering, and social networking [3] [25]. Prominent examples of KGs today include YAGO [18], WordNet [13], Freebase [1], Wikidata [23], the Never-Ending Language Learning system (NELL) [2], and the Google Knowledge Graph [16].

1.2 Knowledge reasoning and link prediction

KGs are often incomplete in practice, usually due to errors in data or limited access to data in their construction. This means that the knowledge a graph embodies does not recognize entities or connections between entities that should exist within it. An important practical consequence of this is that some of these aspects of a KG must be inferred from available information—a task referred to as *knowledge reasoning*, since it is often seen as a process of inferring new conclusions from existing data.

More formally, the general problem of knowledge reasoning is, given an incomplete knowledge graph $\mathcal{G} \subset \mathcal{G}_{\text{gold}}$, to use techniques (often based in machine learning) to infer entities, relations, relation triplets, entity attributes, and more, in order to bring $\mathcal{G}$ closer to $\mathcal{G}_{\text{gold}}$. One prominent component of knowledge reasoning is *link prediction*—also referred to as *relational inference* or *triplet classification*—in which some candidate triplet not in $\mathcal{G}$ is classified as either a positive triplet or a negative triplet [17]. For example, if given the triplets *(Microsoft, IsBasedIn, Seattle)*, *(Seattle, StateLocatedIn, Washington)*, *(Washington, CountryLocatedIn, USA)* in a KG, we would like to predict the missing positive triplet *(Microsoft, IsBasedIn, USA)*.

Multiple classes of methods have been proposed for link prediction. Early methods focused on applying logic rules to the KG to predict the presence or absence of missing relations. While these approaches have the advantage of being semantically intuitive, many of them suffer from scalability and generalizability issues [3]. Other popular methods focus on the applications of statistical reasoning and machine learning to KGs. In this paper, we present a mathematical overview of several methods that fall into this latter category, including relational Markov networks, stochastic relational models, path ranking algorithms, and graph neural networks. While our review is far from exhaustive, it provides a succinct look into the evolution of link prediction methods in the literature and the unique ways in which they exploit graph structure for inference.

2 Approaches

2.1 Relational Markov networks

Historically, most of the work on classification models involves data where individual data points are assumed to be independent and identically distributed. While these assumptions can provide good approximations in some contexts, many data sets involve highly relational data where the classes of different data points are highly correlated. In particular, in settings where graph representations are natural, such strong independence assumptions severely limit the types of relationships between variables that can be modeled.

In an effort to exploit the graph structure inherent in relational data, Taskar *et al.* (2001) apply the framework of *probabilistic relational models* (PRMs) [9, 4], a relational version of Bayesian networks, to classification and clustering of entities over relational data [22]. However, PRMs are directed models, and so in order to form valid probability distributions over variables, the underlying graphical model must be acyclic. This acyclicity constraint severely limits the types of dependencies that can be modeled. As a result, Taskar *et al.* extend this PRM framework to undirected models in the form of *relational Markov networks* (RMNs) applying it first to entity classification [20] and then to link prediction [21]. The authors model link prediction as a special case of entity classification in which relations between entities are treated as first-class citizens.

Relational schema A *relational schema* describes attributes and relations of a set of instance types $\mathcal{E} = \{E_1, ..., E_n\}$. Each type $E \in \mathcal{E}$ is associated with three sets of attributes: content attributes $E.X$, which represent properties of the entity itself; label attributes $E.Y$, which represent the properties to be predicted; and reference attributes $E.R$, which represent entities to which an entity is related.

An instantiation $\mathcal{I}$ of a schema $\mathcal{E}$ specifies the set of entities $\mathcal{I}(E)$ of each entity type $E \in \mathcal{E}$ and the values of all attributes for all of the entities. $\mathcal{I}.X$, $\mathcal{I}.Y$, and $\mathcal{I}.R$ denote the content, label, and reference attributes in the instantiation $\mathcal{I}$, and $\mathcal{I}.x$, $\mathcal{I}.y$, and $\mathcal{I}.r$ denote the values of those attributes. $\mathcal{I}.r$ is called an *instantiation graph* and specifies the set of entities (nodes) and their reference attributes (edges).

Relational Markov networks Let $\mathbf{V}$ denote a set of discrete random variables and $\mathbf{v}$ an assignment of values to $\mathbf{V}$. A Markov network defines a joint distribution over $\mathbf{V}$ consisting of a qualitative component, the undirected dependency graph, and a quantitative component, the set of parameters corresponding to the probability distribution over the nodes in the graph.

In particular, let $G = (\mathbf{V}, E)$ be an undirected graph with a set of cliques $C(G)$. Each $c \in C(G)$ is associated with a set of nodes $\mathbf{V}_c$ and a clique potential $\phi_c(\mathbf{V}_c)$, which is a non-negative function defined on the joint domain of $\mathbf{V}_c$ that represents the compatibility of different assignments to variables. Let $\mathbf{\Phi} = \{\phi_c(\mathbf{V}_c)\}_{c \in C(G)}$. The *Markov network* $(G, \mathbf{\Phi})$ defines a distribution

$$\mathbb{P}(\mathbf{v}) = \frac{1}{Z} \prod_{c \in C(G)} \phi_c(\mathbf{v}_c) \tag{1}$$

where Z is the partition function given by $Z = \sum_{\mathbf{v}'} \phi_c(\mathbf{v}'_c)$.

We can similarly define conditional Markov networks as follows. Let $\mathbf{X}$ be a set of random variables on which we condition and $\mathbf{Y}$ be a set of target random variables. A

conditional Markov network $(G, \mathbf{\Phi})$ defines the distribution

$$\mathbb{P}(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in C(G)} \phi_c(\mathbf{x}_c, \mathbf{y}_c) \tag{2}$$

where $Z(\mathbf{x})$ is the partition function given by $Z(\mathbf{x}) = \sum_{\mathbf{y}'} \phi_c(\mathbf{x}_c, \mathbf{y}'_c)$.

Before finally defining relational Markov networks, we must define the notion of *relational clique templates*, which are used to specify cliques of interest for our model. In particular, a relational clique template $C = (\mathbf{F}, \mathbf{W}, \mathbf{S})$ consists of $\mathbf{F} = \{F_i\}$, a set of entity variables; $\mathbf{W}(\mathbf{F}.\mathbf{R})$, a Boolean formula using conditions of the form $F_i.R_j = F_k.R_l$; and a selected subset $\mathbf{F}.\mathbf{S} \subseteq \mathbf{F}.\mathbf{X} \cup \mathbf{F}.\mathbf{Y}$ of content and label attributes in $\mathbf{F}$. This clique template allows us to query a specific set of cliques in our instantiation graph

$$C(\mathcal{I}) \coloneqq \{c = \mathbf{f}.\mathbf{S} : \mathbf{f} \in \mathcal{I}(\mathbf{F}) \wedge \mathbf{W}(.\mathbf{r})\} \tag{3}$$

where $\mathbf{f}$ is a tuple of entities $\{f_i\}$ in which each f_i is of type $E(F_i)$, $\mathcal{I}(\mathbf{F}) = \mathcal{I}(E(F_1)) \times \cdots \times \mathcal{I}(E(F_n))$ denotes the cross-product of entities in the instantiation, the clause $\mathbf{W}(\mathbf{f}.\mathbf{r})$ ensures that the entities are related to each other in the desired way, and $\mathbf{f}.\mathbf{S}$ extracts the desired attributes from the entities which match the query.

Now, a *relational Markov network* (RMN) $\mathcal{M} = (\mathbf{C}, \mathbf{\Phi})$ specifies a set of clique templates $\mathbf{C}$ and corresponding potentials $\mathbf{\Phi} = \{\phi_C\}_{C \in \mathbf{C}}$ to define a conditional probability distribution

$$\mathbb{P}(\mathcal{I}.\mathbf{y}|\mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r}) = \frac{1}{Z(\mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r})} \prod_{C \in \mathbf{C}} \prod_{c \in C(\mathcal{I})} \phi_C(\mathcal{I}.\mathbf{x}_c, \mathcal{I}.\mathbf{y}_c) \tag{4}$$

where $Z(\mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r})$ is the normalizing partition function

$$Z(\mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r}) = \sum_{\mathcal{I}.\mathbf{y}'} \phi_C(\mathcal{I}.\mathbf{x}_c, \mathcal{I}.\mathbf{y}'_c) \tag{5}$$

Estimating parameters Given a fixed set of relational clique templates, the task of learning an RMN reduces to estimating the weights $\mathbf{w}$ for clique potentials $\mathbf{\Phi}$. In particular, given an instantiation $\mathcal{I}$ of the schema, an RMN $\mathcal{M}$ produces an unrolled Markov network over the attributes of entities in $\mathcal{I}$. The cliques in the unrolled graph are determined by the clique templates $C \in \mathbf{C}$, where each $c \in C$ shares a clique potential ϕ_C.

A specific choice of parameters $\mathbf{w}$ specifies a particular instance of an RMN, which induces a probability distribution $\mathbb{P}_{\mathbf{w}}$ over the unrolled Markov network. Given some data set D, the likelihood of the data is $\prod_{d \in D} \mathbb{P}_{\mathbf{w}}(\mathbf{y}_d|\mathbf{x}_d)$, which can be optimized by gradient descent over $\mathbf{w}$. Computing this gradient requires inference over the unrolled Markov network, which can be intractable over large networks. Therefore, Taskar *et al.* suggest the use of belief propagation [20].

2.2 Stochastic relational models

Yu et al. approached the task of relational inference by modeling the relationships between all pairs of entities in a KG using multiple entity-wise Gaussian processes in a family of models they call *stochastic relational models* (SRMs) [24]. In essence, SRMs aim to model the observed relations in a KG using a latent variable model whose hyperparameters can be learned through variational inference. These optimal hyperparameters can then be used to

calculate the marginal posterior probabilities for relations between new pairs of entities in the KG.

The authors first consider pairwise, asymmetric links $r \in \mathcal{R}$ between entities $u \in \mathcal{U}$ and $v \in \mathcal{V}$, noting that the two sets $\mathcal{U}, \mathcal{V}$ can be the same or different, as well that u and v may be used to represent the identity and/or attribute vectors of entities in their respective sets. Let $r_{i,n} \equiv r(u_i, v_n)$ denote a relation of type r between entities u_i and v_n. The first key assumption of SRMs is that the observable links r are derived as local measurements of a real-valued latent relational function $t : \mathcal{U} \times \mathcal{V} \to \mathbb{R}$ and that each link $r_{i,n}$ is solely dependent on its latent value $t_{i,n}$, modeled by the likelihood $\mathbb{P}(r_{i,n}|t_{i,n})$. The second key assumption is that the latent relational function t is generated by a tensor interaction of two independent, entity-specific Gaussian processes, one acting on $\mathcal{U}$ that is parameterized by θ_Σ and the other acting on $\mathcal{V}$ that is parameterized by θ_Ω. Together, these Gaussian processes help define a Bayesian prior $\mathbb{P}(t|\theta)$ for the latent variable t. The evidence under this prior is then defined as:

$$\mathbb{P}(\mathbf{R}_\mathbb{I}|\theta) = \int \prod_{(i,n)\in\mathbb{I}} \mathbb{P}(r_{i,n}|t_{i,n})\mathbb{P}(t|\theta)dt, \quad \theta = \{\theta_\Sigma, \theta_\Omega\} \tag{6}$$

where $\mathbb{I}$ is the index set of entity pairs having observed links such that $\mathbf{R}_\mathbb{I} = \{r_{i,n}\}_{(i,n)\in\mathbb{I}}$.

The authors propose using a family of stochastic relational processes to calculate the prior $\mathbb{P}(t|\theta)$ in Eq. (6) such that $t : \mathcal{U} \times \mathcal{V} \to \mathbb{R}$ takes the form

$$t(u, v) = \frac{1}{\sqrt{d}} \sum_{k=1}^{d} f_k(u) g_k(v) \tag{7}$$

where $f_k(u) \overset{iid}{\sim} \mathcal{GP}(0, \Sigma), g_k(v) \overset{iid}{\sim} \mathcal{GP}(0, \Omega)$, and d represents the degrees of freedom. To calculate the likelihood $\mathbb{P}(r_{i,n}|t_{i,n})$, the authors state that multiple choices exist depending on the inference problem. For binary classification relation inference (i.e., is there a link between entities u and v or not?), they suggest using $\mathbb{P}(r_{i,n}|t_{i,n}) = \Phi(r_{i,n}t_{i,n})$ where $\Phi(\cdot)$ is a cumulative normal function and $r_{i,n} \in \{-1, +1\}$.

The parameters θ are then learned through expectation-maximization. Let $\mathbf{F} = \{f_{i,k}\}, \mathbf{G} = \{g_{n,k}\}, \mathbf{f}_k = [f_{1,k}, ..., f_{N,k}]^T$, and $\mathbf{g}_k = [g_{1,k}, ..., g_{M,k}]^T$, where $f_{i,k} = f_k(u_i)$ and $g_{n,k} = g_n(v_k)$. The E-step involves calculating the posterior probability $\mathbb{P}(\mathbf{F}, \mathbf{G}|\mathbf{R}_\mathbb{I}, \theta)$ which is proportional to the joint distribution of the complete data and can be factorized as:

$$\mathbb{P}(\mathbf{F}, \mathbf{G}, \mathbf{R}_\mathbb{I}|\theta) \propto \prod_{(i,n)\in\mathbb{I}} \mathbb{P}\left(r_{i,n} \middle| \frac{\sum_{k=1}^{d} f_{i,k} g_{n,k}}{\sqrt{d}}\right) \exp\left\{ -\frac{1}{2} \sum_{k=1}^{d} [\mathbf{f}_k^T \Sigma^{-1} \mathbf{f}_k + \mathbf{g}_k^T \Omega^{-1} \mathbf{g}_k] \right\} \tag{8}$$

Exact inference in (8) is intractable due to the coupling of f and g in the the likelihood, so the authors choose to calculate the value using a Laplacian approximation, which approximates $\mathbb{P}(\mathbf{F}, \mathbf{G}|\mathbf{R}_\mathbb{I}, \theta)$ using a multivariate normal distribution $q(\mathbf{F}, \mathbf{G}|\beta)$ with sufficient statistics β. The means of β are computed through solving the following minimization problem,

$$\{\mathbf{F}^*, \mathbf{G}^*\} = \arg\min_{\{\mathbf{F},\mathbf{G}\}} \left\{ J(\mathbf{F}, \mathbf{G}) = -\log \mathbb{P}(\mathbf{R}_\mathbb{I}, \mathbf{F}, \mathbf{G}|\theta) \right\} \tag{9}$$

(9) can be solved using the conjugate gradient method, with gradients calculated as

$$\frac{\partial J(\mathbf{F}, \mathbf{G})}{\partial \mathbf{F}} = \frac{1}{\sqrt{d}} \mathbf{S}\mathbf{G} + \Sigma^{-1}\mathbf{F}, \quad \frac{\partial J(\mathbf{F}, \mathbf{G})}{\partial \mathbf{G}} = \frac{1}{\sqrt{d}} \mathbf{S}^T\mathbf{F} + \Omega^{-1}\mathbf{G} \tag{10}$$

where $\mathbf{S} \in \mathbb{R}^{N \times M}$ have elements

$$s_{i,n} = \begin{cases} \frac{\partial[-\log \mathbb{P}(r_{i,n}|t_{i,n})]}{\partial t_{i,n}} \text{ if } (i,n) \in \mathbb{I} \\ 0, \text{ otherwise} \end{cases}$$

To solve for the covariance matrices, the authors first assume that there exist disjoint groups of latent variables, which allow the distribution q to be factorized as $q(\mathbf{F}, \mathbf{G}|\beta) = \prod_{k=1}^{d} q(\mathbf{f}_k|\mathbf{f}_k^*, \Sigma_k) q(\mathbf{g}_k|\mathbf{g}_k^*, \Omega_k)$. This allows for the inverse Hessian of $J(\mathbf{F}, \mathbf{G})$ to be calculated for each group separately, yielding the covariance matrices:

$$\Sigma_k = (\Phi^{(k)} + \Sigma^{-1})^{-1}, \quad \text{with} \quad \phi_{i,i}^{(k)} = \sum_{n:(i,n)\in\mathbb{I}} \frac{\zeta_{i,n} g_{n,k}^2}{d}, \quad \phi_{i,j}^{(k)} = 0 \tag{11}$$

$$\Omega_k = (\Psi^{(k)} + \Sigma^{-1})^{-1}, \quad \text{with} \quad \psi_{n,n}^{(k)} = \sum_{i:(i,n)\in\mathbb{I}} \frac{\zeta_{i,n} f_{i,k}^2}{d}, \quad \psi_{n,m}^{(k)} = 0 \tag{12}$$

where $\zeta_{i,n} = \frac{\partial^2[-\log \mathbb{P}(r_{i,n}|t_{i,n})]}{\partial t_{i,n}^2}$.

The M-step involves assigning a hyper prior $\mathbb{P}(\theta|\alpha)$ and estimating θ by maximizing a penalized marginal likelihood:

$$\theta^* = \arg \max_{\theta=\{\theta_\Sigma, \theta_\Omega\}} \left\{ \log \int_{\mathbf{G}} \int_{\mathbf{F}} \mathbb{P}(\mathbf{R}_\mathbb{I}, \mathbf{F}, \mathbf{G}|\theta) d\mathbf{F} d\mathbf{G} + \log \mathbb{P}(\theta|\alpha) \right\} \tag{13}$$

The authors choose to assign both of these hyper priors as conjugated inverse-Wishart distributions parameterized by λd and Σ_0 or Ω_0, appropriately. Then, the hyperparameters Σ and Ω are updated by maximizing the expected log-likelihood of the complete data

$$\max_{\{\Sigma,\Omega\}} \mathbb{E}_{q(\mathbf{F},\mathbf{G}|\beta)}[\log \mathbb{P}(\mathbf{R}_\mathbb{I}, \mathbf{F}, \mathbf{G}|\Sigma, \Omega)] + \log \mathbb{P}(\Sigma|\lambda d, \Sigma_0) + \log \mathbb{P}(\Omega|\lambda d, \Omega_0) \tag{14}$$

This has an analytical solution of

$$\Sigma = \frac{\lambda\Sigma_0 + \frac{1}{d}\sum_{k=1}^{d}(\mathbf{f}_k^* \mathbf{f}_k^{*T} + \Sigma_k)}{\lambda + 1}, \quad \Omega = \frac{\lambda\Omega_0 + \frac{1}{d}\sum_{k=1}^{d}(\mathbf{g}_k^* \mathbf{g}_k^{*T} + \Omega_k)}{\lambda + 1} \tag{15}$$

In summary, the algorithm iterates between solving Eqs. (9), (11), (12) in the E-step and (13) in the M-step until convergence. Then $\mathbb{P}(r_{i,n}|t_{i,n}^*)$ are used to make predictions, where t^* is computed from $\mathbf{F}^*$ and $\mathbf{G}^*$ [24].

2.3 Path ranking algorithms

Lao and Cohen originally introduced the *path ranking algorithm* (PRA) as a method for answering what they refer to as *typed proximity queries* over a knowledge graph [10]. In such queries, a user provides as input a set of query nodes and a relation type, and receives as output a ranked list of nodes ordered by probability of connection to the query node via the relation type. The authors were specifically interested in queries over knowledge graphs of published biomedical literature, such as venue recommendation (i.e., recommending a venue to publish a new research paper), reference recommendation (i.e., recommending relevant citations for a new paper), expert finding (i.e., finding a domain expert for a particular

topic), and gene recommendation (i.e., predicting what genes a researcher will publish on over the next year).

The PRA is composed of three components: (1) feature selection through the enumeration of all possible relation paths of length l connecting query and target nodes; (2) feature computation via recursively computing the probabilities that a random walker would arrive at a target node from the source node via a path of the types enumerated in (1); and (3) relation-specific classification based on the features computed in (2).

To describe the algorithm in further detail, we first specify a few definitions. Let r be a binary relation, such that we denote $r(e, e')$ if e and e' are related by r. Let $dom(r)$ denote the domain of the relation and let $range(r)$ denote its range. A *relation path* (or *path*, for short) P, is then a sequence of relations $r_1, ..., r_\ell$, such that $\forall 1 < i < \ell - 1, range(r_i) = dom(r_{i+1})$.

Feature selection in Lao and Cohen's original PRA consists of enumerating all possible relation paths between a specified query and target node of length at most ℓ via a random walk. Let $P_\ell = \{P_1, ..., P_n\}$ represent this set of paths. These path types represent features that are likely to be useful in predicting new instances of the relation represented by the query and target node pairs. Each of these enumerated relation paths then takes a value computed via the path-constrained random walk described below.

For any relation path $P = R_1, ..., R_\ell$ and a query node $s \in dom(P)$, a path-constrained random walk defines a distribution $h_{s,P}$ over a target node e in the following manner. If P is the empty path, define

$$h_{s,P}(e) = \begin{cases} 1, \text{ if } e = s \\ 0, \text{ otherwise} \end{cases} \tag{16}$$

If P is not empty, let $P' = R_1, ..., R_{\ell-1}$, and define the distribution

$$h_{s,P}(e) = \sum_{e' \in range(P')} h_{s,P'}(e') \cdot \mathbb{P}(e|e'; R_l) \tag{17}$$

where $\mathbb{P}(e|e'; R_l) = \frac{\mathbf{1}\{R_\ell(e', e)\}}{|R_\ell(e', \cdot)|}$ is the probability of reaching node e from node e' with a one-step random walk with edge type R_ℓ. From this definition, we can see that $h_{s,P}(e) = \mathbb{P}(e|s, P)$, or, the probability that a random walker would reach node e from s by following a path of the relation types specified in P.

This random walk is repeated for all paths in P_ℓ, resulting in features with values $\{h_{s,P_i}(e)\}_{i=1}^n$. The idea is then to calculate a score for e based on its relation to s as a weighted linear combination of the features, i.e.

$$score(e; s) = \sum_{i=1}^{n} h_{s,P_i}(e)\theta_i \tag{18}$$

If done for multiple target nodes, this score can be used to rank them as answers to the query defined by s and a relation r.

From this point, the parameters θ must be learned for each $r \in \mathcal{R}$, which the authors pursue via logistic regression. Given a relation r and a set of node pairs $\{s_i, e_i\}$ for which we know whether or not $r(s_i, e_i)$ holds, a training dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$ can be created such that $\mathbf{x}_i$ is a vector of all the path features $\{h_{s_i,P_j}(e_i)\}_{j=1}^n$ and y_i is a boolean indicator of whether or not $r(s_i, e_i)$ holds. $\mathcal{D}$ is used to train a logistic function to predict the

conditional probability $\mathbb{P}(y|x;\theta)$, in effect learning the parameter vector θ, by maximizing the L_2-regularized objective function:

$$J(\theta) = \sum_i [y_i \log p_i + (1 - y_i) \log(1 - p_i)] - \lambda \|\theta\|_2 \tag{19}$$

where $p_i = \mathbb{P}(y_i = 1|\mathbf{x}_i;\theta) = \frac{\exp(\theta^T \mathbf{x}_i)}{1+\exp(\theta^T \mathbf{x}_i)}$. The authors note that other cost functions could be used to fit different classifiers to this data, such as negative hinge loss in the case of support vector machines or negative exponential loss for boosting; however, almost all of the literature covered in this review uses the binomial log-likelihood. Such a classifier is trained for each relation $r \in \mathcal{R}$, whose parameters can be used to calculate scores for each pair of query and target nodes in the KG as defined in (18) for the proximity query question originally posed by the authors. Alternatively, these classifiers can be used directly for relational inference between nodes in the KG.

2.3.1 PRA extensions

Since Lao and Cohen's publication of the PRA in [10], numerous extensions of the algorithm have been created that attempt to improve its performance and scalability for relational inference, as well as to overcome certain incapacities it possesses. One issue with the original algorithm is that it can quickly become impractical to enumerate all relation paths between two nodes in a KG as the number of possible edge types grows, even when the paths are constrained to be of length at most l. Lao et al. approach this issue by modifying the path generation procedure in PRA to produce a smaller and more informative set of relation paths (features) which are more useful for relational inference [11]. Another major issue with PRA is that if there is no path connecting two nodes in the graph, then the algorithm cannot predict any relation instance between them. Gardner et al. address this deficiency by allowing the path-constrained random walk in the feature computation step to follow a different outgoing edge at any point along the path, so long as the outgoing edge is semantically similar (in terms of vector space representations) to the one in the original path [6]. Other algorithms have been developed as well, such as Subgraph Feature Extraction [5] and Hierarchical Random-walk Inference algorithm [12], which combine a global PRA procedure with local procedures that create more expressive features from the neighborhoods of entity pairs within a graph. These variations all show major predictive and computational performance improvements over the original PRA presented here, but we omit a more thorough explanation of them in this paper.

2.4 Message passing & graph neural networks

Message-passing algorithms for inference on graphical models were first introduced by Judea Pearl in 1982 [14] and have since been applied to many fields where graphs provide canonical representations, such as artificial intelligence, information theory, and physics. However, inference using message-passing algorithms can only be computed exactly on trees; in general graphs, they can only provide approximations. In an effort to take advantage of modern advancements in deep learning and to further generalize the task of inference over graphical models, Gilmer *et al.* (2017) formalized the notion of *graph neural networks* (GNNs), originally calling them *message passing neural networks* [7].

A GNN takes as input a graph $G = (V, E)$ which can be directed or undirected. Each node $i \in V$ is associated with a feature vector $\mathbf{x}_i$, and optionally each edge $(i, j) \in E$ can

also be associated with a feature vector $\mathbf{x}_{(i,j)}$. Each feature vector is associated with a hidden representation, $\mathbf{h}_i$ for nodes and $\mathbf{h}_{(i,j)}$ for edges. Initially, $\mathbf{h}_i = \mathbf{x}_i$, and "messages" are propagated along edges in the graph as follows

$$\begin{aligned} \mathbf{h}_{(i,j)} &= f_{\text{edge}}(\mathbf{h}_i, \mathbf{h}_j, \mathbf{x}_{(i,j)}) \\ \mathbf{h}'_i &= f_{\text{node}}(\mathbf{h}_i, \sum_{j \in \mathcal{N}_i} \mathbf{h}_{(j,i)}, \mathbf{x}_i) \end{aligned} \tag{20}$$

where $\mathcal{N}_i$ is the set of nodes with edges directed into node i, and f_{edge} and f_{node} are learnable functions, typically neural networks.

Kipf approaches the task of link prediction using *graph auto-encoders* (GAEs) [8]. That is, a graph convolutional network (GCN) encoder which takes as input the data $\mathbf{X}$ and graph adjacency matrix $\mathbf{A}$ and outputs a matrix $\mathbf{Z}$ of hidden representations. In particular, Kipf uses a two-layer GCN encoder of the form

$$\mathbf{Z} = \hat{\mathbf{A}}\text{ReLU}(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}_0)\mathbf{W}_1 \tag{21}$$

where $\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix ($\mathbf{D}$ is the diagonal degree matrix, $D_{i,i} = \sum_j A_{i,j}$), and $\mathbf{W}_0 \in \mathrm{R}^{d_{\text{in}} \times d_{\text{hid}}}$ and $\mathbf{W}_1 \in \mathrm{R}^{d_{\text{hid}} \times d_{\text{emb}}}$ are trainable parameter matrices.

The task of the decoder is then to reconstruct the adjacency matrix $\mathbf{A}$ from $\mathbf{Z}$. This is done by introducing a scoring function $s(\mathbf{z}_i, \mathbf{z}_j)$ that is designed to assign high scores to nodes which should be connected and low scores to nodes which should not be connected. In particular, Kipf uses an inner-product scoring function $s(\mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i^\top \mathbf{z}_j$ such that the reconstructed adjacency matrix A' is computed as

$$A' = \sigma(\mathbf{Z}\mathbf{Z}^\top) \tag{22}$$

where σ is the logistic sigmoid function [8].

The GAE can then be trained by optimizing the cross-entropy loss

$$\mathcal{L} = -\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} A_{i,j}\log\sigma(s(\mathbf{z}_i, \mathbf{z}_j)) + (1 - A_{i,j})\log(1 - \sigma(s(\mathbf{z}_i, \mathbf{z}_j))) \tag{23}$$

When applied to the Cora, Citeseer, and Pubmed data sets, GAE and variational GAE models taking advantage of node features produce better average precision and area under the ROC curve scores compared to two popular baselines SpectralEmbedding [19] and DeepWalk [15], demonstrating the power of GNNs for performing inference over graphs.

3 Conclusion

In this paper, we have presented a brief overview of several methods for conducting link prediction over knowledge graphs. Each of these methods utilizes the topology of an incomplete KG, $\mathcal{G}$, in distinct ways to infer probabilistically the presence or absence of proposed triplets in the corresponding complete KG, $\mathcal{G}_{\text{gold}}$. This survey is far from exhaustive, and we have notably omitted many popular methods which focus on projecting the knowledge graph into continuous vector space and conducting inference on these embeddings. From this, it should be clear that link prediction in KGs has been and remains a rich field of research, which will likely continue to evolve with current advances in machine and deep learning.

4 References

[1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM Sigmod International Conference on Management of Data*, pages 1247–1250, 2008.

[2] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*, 2010.

[3] Xiaojun Chen, Shengbin Jia, and Yang Xiang. A review: Knowledge reasoning over knowledge graph. *Expert Systems with Applications*, 141:112948, 2020.

[4] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *IJCAI*, volume 99, pages 1300–1309, 1999.

[5] Matt Gardner and Tom Mitchell. Efficient and expressive knowledge base completion using subgraph feature extraction. In *Conference on Empirical Methods in Natural Language Processing*, pages 1488–1498, 2015.

[6] Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. Incorporating vector space similarity in random walk inference over knowledge bases. In *Conference on Empirical Methods in Natural Language Processing*, pages 397–406, 2014.

[7] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry, 2017.

[8] Thomas N Kipf. Deep learning with graph-structured representations. 2020.

[9] Daphne Koller and Avi Pfeffer. Probabilistic frame-based systems. In *AAAI/IAAI*, pages 580–587, 1998.

[10] Ni Lao and William Cohen. Relational retrieval using a combination of path-constrained random walks. *Mach Learn*, 81:53–67, 2010.

[11] Ni Lao, Tom Mitchell, and William Cohen. Random walk inference and learning in a large scale knowledge base. In *Conference on Empirical Methods in Natural Language Processing*, pages 529–539, 2011.

[12] Qiao Liu, Liuyi Jiang, Minghao Han, Yao Liu, and Zhinguang Qin. Hierarchical random walk inference in knowledge graphs. *ACM*, 2016.

[13] G.A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38:39–41, 1995.

[14] Judea Pearl. *Reverend Bayes on inference engines: A distributed hierarchical approach.* Cognitive Systems Laboratory, School of Engineering and Applied Science . . . , 1982.

[15] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[16] Amit Singhal. Introducing the knowledge graph: things, not strings. https://blog.google/products/search/introducing-knowledge-graph-things-not/, 2012. Accessed 2021-12-16.

[17] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934, 2013.

[18] F.M. Suchanek, G. Kasneci, and G. Weikum. Yago: A large ontology from wikipedia and wordnet. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6:203–217, 2008.

[19] Lei Tang and Huan Liu. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery*, 23(3):447–478, 2011.

[20] Ben Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. *arXiv preprint arXiv:1301.0604*, 2002.

[21] Ben Taskar, Ming-Fai Wong, Pieter Abbeel, and Daphne Koller. Link prediction in relational data. *Advances in neural information processing systems*, 16:659–666, 2003.

[22] Benjamin Taskar, Eran Segal, and Daphne Koller. Probabilistic classification and clustering in relational data. In *International joint conference on artificial intelligence*, volume 17, pages 870–878. Lawrence Erlbaum Associates LTD, 2001.

[23] Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57:78–85, 2014.

[24] Kai Yu, Wei Chu, Shipeng Yu, Volker Tresp, and Zhao Xu. Stochastic relational models for discriminative link prediction. In *NIPS*, 2006.

[25] Xiaohan Zou. A survey on application of knowledge graph.