Towards Uniform Semantic Interpretation

Dimitar Hrisimirov Popov

Bulgarian Academy of Sciences, Institute of Philosophy and Sociology

dimiturpopov1990@gmail.com

Семантична интерпретация чрез еднородно унифициране

Димитър Хрисимиров Попов

Българска академия на науките, Институт по философия и социология

dimiturpopov1990@gmail.com

***Abstract**: Artificial Intelligence methodologies for natural language processing has steadily converged towards heavy deployment of Large Language Models. The underlying operational principle of these models relies on mining massive sets of text corpuses, where the semantic elicitation between the entities of the text is interpreted as statistical interdependency (correlation) between words located close to each other in a sentence. The problem with this approach is that probable semantic connection could only be established, if such connections exist in the already mentioned large text corpuses, furthermore, in order for models further to generalize over unseen examples, this plethora of training examples must be provided, in a vastly jumbled and diverse texts, which is inherently problematic, if the model has to be updated with new correlations, since new text corpuses has to be carefully selected or synthetically generated in order not to strongly weaken initial correlations. As a rule of thumb, in contrast to the artificial large language models, humans, need but a few examples, in order to embed a possible semantic model that governs any new concept, independently of the informational structure of the examples, be that text, images, graphs, etc. Although, human cognition is a constantly proliferated domain of research, most researchers would agree that humans are able to elicit semantic, not only by learning interdependencies between entities, but also semantic relationship rules that governs those entities. This knowledge about the rules and entities is then constantly updated throughout our life. This paper explores one possibility of creating such hybrid computational model, where semantic relationships rules are created out of graph structured data and are used to improve*

semantic interpretation in natural language processing by unification of two learning paradigms on graph textual representation, to improve semantic interpretation.

***Keywords**: Artificial Intelligence, natural language processing, hybrid models, graph, graph neural, network.*

***Резюме**: Постепенното подобряване на методите за изкуствен интелект за обработка на естествен език доведе до създаването и силното използване на така наречените огромни езикови модели. Тези модели разчитат на парадигми за извличане на знание от огромни масиви от текстови данни, където семантичните връзки между различните обекти се осъществява на база на калкулация на процент статистическа обвързаност (корелация) между позицията на думите в дадено изречение. Проблемите които тези методи индуцират при тяхното обучение и работа са обвързани основно със силната зависимост, че семантичните връзки между обектите, трябва да присъстват в тренировъчните данни и че потенциална генерализация над непознати данни е възможна само, след като огромна база от текстове представящи различни семантични взаимообвързаности между обектите е използвана за обучение. Това е проблематично, защото при нужда за преобучение на модела е необходимо нови масиви да бъдат внимателно подбрани или синтетично генерирани за да се избегне отслабването на корелационни връзки между вече съществуващите обекти. От друга страна хората се нуждаят само от няколко примера за да генерират семантичен модел, който определя връзките между различни типове обекти, било то представени в текстове, изображения, схеми или други. Въпреки че изследователското поле на разглеждане на човешкия интелект и когнитивност търпи постоянно обогатяване и развитие, повечето специалисти се съгласяват, че хората не научават просто корелационни зависимости, обвързаности или еднаквост между думите, а също и правилата които формализират връзките между обектите. В тази статия ще представи един възможен хибриден изчислителен модел, в който семантичните правила между обектите се извличат чрез графово представяне на данни и тези правила се използват за подобряване на семантичната интерпретация при обработка на естествен език, чрез унифициране на два*

метода за обучение върху граф и добиване на еднородност на интерпретацията по този начин.

***Ключови думи**: Изкуствен интелект, обработка на естествен език, хибридни модели, граф, графови невронни мрежи.*

The aim of this paper is to create a theoretical outlier of hybris artificial intelligence system, which uses both symbolic logic and sub-symbolic methodologies for learning to create unified approach for discovery of semantic correlation of entities based on similarity measures and their relationship rules that governs them. Since this is extremely broad formalization, we will concentrate our effort on a particular data structure, which has powerful enough representation and could be used to transform comfortably enough, other representations like text, images, tables, namely graphs.

Why graphs? Graphs are non-linear data structures and as such has expressive power to capture almost every hidden relation from another type of data. For example, in natural language process a textual representation of character string (e.g. "*Dimitar lives in Ravda*", (1)) could be transformed as graph that contains set of related entities and their relation. If we consider the example in (1) a graph representation would look like the figure 1 below. In the simple schematics entities *{Dimitar, Ravda}* are members of entity set containing all the entities for given graph. In graph theory these are called nodes of edges of a graph. The link between the two nodes is the semantic relation between two nodes, in the simple example on Figure 1, the relation is named *lives_in* . In a graph, relations are called vertexes or links, vertexes could be directed, if they are depicted with an arrow.

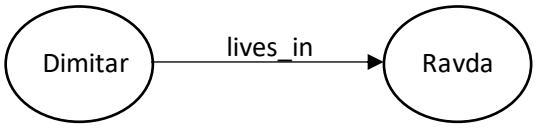

Figure 1. Graph with two nodes representing semantic relation: lives_in

This depicting arrow carries information about the orientation of the edges. There are various graph relationship mining methods, which formalize mathematical model to discover relationships in a complex graph structure. Based on those methods and the represented graphs, one may wish to discover what are the relation to yet unidentified link in the graph or classified a node label altogether. In figure 2 we could see a graph, that has one unlabeled node connected with one unlabeled vertex to the node Ravda.

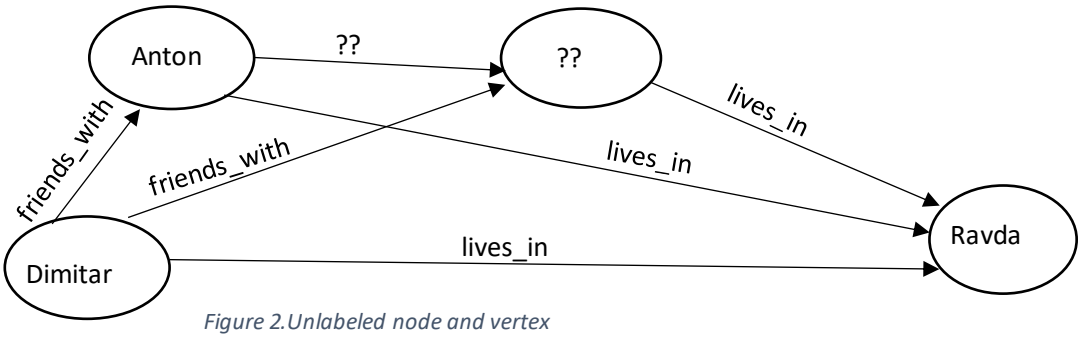

Figure 2.Unlabeled node and vertex

Predicting node labels and missing links in graphs is exactly how semantic interpretation is built upon graphs. Looking up at the whole graphs as computational unit each relation and entities represent an information about certain fact that exist as interconnected entities inside the mesh. Machine learning algorithms which mine knowledge from graph, investigate the creation of models that learns and operate on latent representations of entities and relations. These methods condense each entity's (node's) neighborhood connectivity pattern into entity specific low-dimensional embedding, which can then be used to predict missing edges based on some similarity evaluation like Euclidian distance in the new low-dimension space {1}. On figure 2, the missing link between nodes *{Dimitar → ??}*, will be embedded into the proximity of two other vertexes *friends_with*, the embedding algorithm relies on statistical evaluation to predict if the missing link could be classified as another *friends_with* vertex. Such node embedding approaches do not explicitly capture compositional logical rules underlying a provided graph, and they are limited to transductive setting, where full set of entities must be known during training, this limitation is somehow in parallel with the large language model limitations. Towards building unified approach for capturing both relational semantics between entities and the semantic classification definition of a entity, we construct a solution oriented towards first capturing the node embedding for identifying both similarities between nodes and vertexes and then capture the rule governing the semantic relationships. These elicited rules, unlike with the embedding approach, are naturally

inductive and can generalize over unseen entities and graphs after training. To formalize our unification method, we will take advantage of one particular type of graph relational data representation named Knowledge Graph. This representation contains a collection of interlinked descriptions of entities, the same entities which semantic relations and governing rule we would like to express. Construction of Knowledge Graphs from another type of data representation is a whole topic on its own, but a short conceptualization of the formalism specifics of the graphs and the rules that could be extracted will be displayed in the following section.

Knowledge Graph(KG). Consists of two sets $K = (E, F)$ where *E* is the set of entities and *F* is the set of facts. A fact in knowledge graph is represented as a triple $k = (s, P, o)$, which means that the subject entity ***s*** is related to the object entity ***o*** via predicate *P. P(s, o)* is true if the fact *k = (s, P, o)*, exists in *K.*

Rules. Let *R = {P}* denote the set of all the predicates. Then for each closed path triplets in the graph, the form of a rile *r* is:

$$P_1(x, z_1) \wedge P_2(z_1, z_2) \wedge \ldots \wedge P_n(z_{n-1}, y) \rightarrow P_0(x, y) \quad (1)$$

The equation above has an inductive logical form meaning that every new discovered fact would contribute to the above form by adding a predicate form P_i *(s, o),* named atom, to the logical expression. In (1) *x, y,* z_i are variables entities, *s, o* are called subject and object argument for P_i, *n* is the length of the rule *r,* and the length of the rule is *n+1.* We also identify a head which is the atom after the implication sign ($\ldots \rightarrow P_0(x, y)$) and tail which is the atomics' conjunction before the implication $P_1(x, z_1) \wedge P_2(z_1, z_2) \wedge \ldots \wedge P_n(z_{n-1}, y)$. The rule *r* is closed path if the sequence of predicates in the rule's body forms a path from the subject argument to the object argument of the head. For example, if we take the relations in figure 3, we could create a close-path rule based on the triplet relation {Dimitar, married_to, Daniela},

$$\exists Y(X, married\_to, Y) \wedge (Y, lives\_in, Z) \rightarrow (X, lives\_in\ Z). \quad (2)$$

Using this example from figure 3, we can predict the relation *{Dimitar, lives_in, Ravda}*

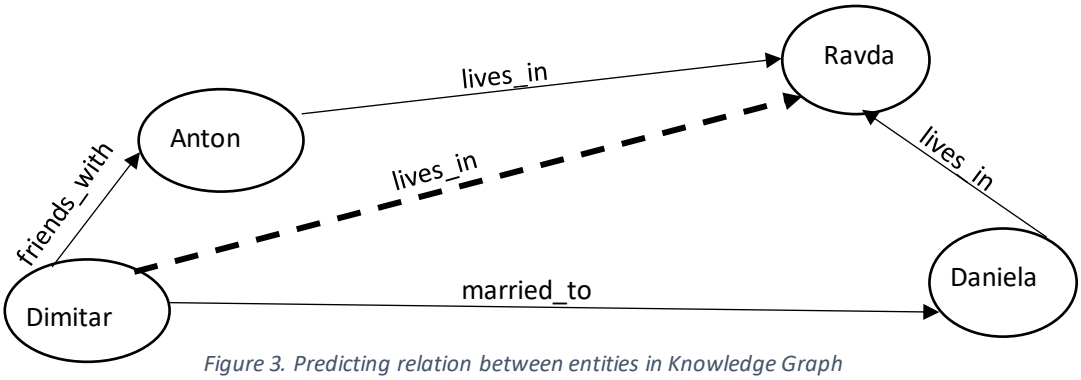

Figure 3. Predicting relation between entities in Knowledge Graph

While the embedding-based methods encode entity specific neighborhood information into an embedding, these logical rule capture entity-independent relationship semantics. If we generate enough semantic relationships in the inductive form, the question is can we unify the rules with some other type of logical expression extracted from the graph, that carry information not only about the relationship of the nodes, but specifics about a given node. For example, if we have a Knowledge Graph that represent information about animal taxonomy and attributes of animals, and we would like to represent information about the different sizes and colours of every single bird we must encode this information as node features, node features are parameters that introduce deep dimensionality in graphs, however, they are not common in Knowledge Graphs, since the idea of Knowledge Graphs are to capture general knowledge about entities, but not individual properties for each separate entity. The illustration below shows how a Knowledge Graph, captures the knowledge about feather colorization among swans.

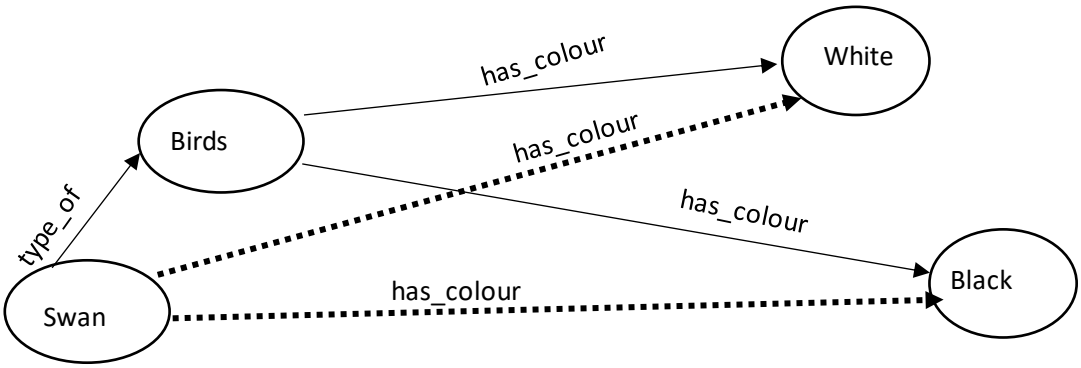

Figure 4. Knowledge Graph representing some facts about swans.

The Knowledge Graph on figure has in it encoded semantic relation between the node *Swan* and nodes *White, Black,* we could create inductive rule as in (2) and predict that also that node *Birds*

is semantically connected with *Black, White*, since it is enclosed in a *close-path* relation triplet with *Swan:*

$$\exists Y(Swan, type\_of, Birds) \wedge (Birds, has\_colour, White) \rightarrow (Swan, has\_colour, White).$$

$$\exists Y(Swan, type\_of, Birds) \wedge (Birds, has\_colour, Black) \rightarrow (Swan, has\_colour, Black). \quad (3)$$

The union of these two extracted rules is in the form

$(Swan, has\_colour, White) \wedge (Swans, has\_colour, Black)$ this expression is semantically entailed to True if both of the facts are present in the Knowledge Graph. The problem with rule discovery based on triplets is that, we could discover rules which logically follows the shown pattern, meaning they are logically valid, but not sound. In figure 4 we get one more rule which changes the form of (3) to

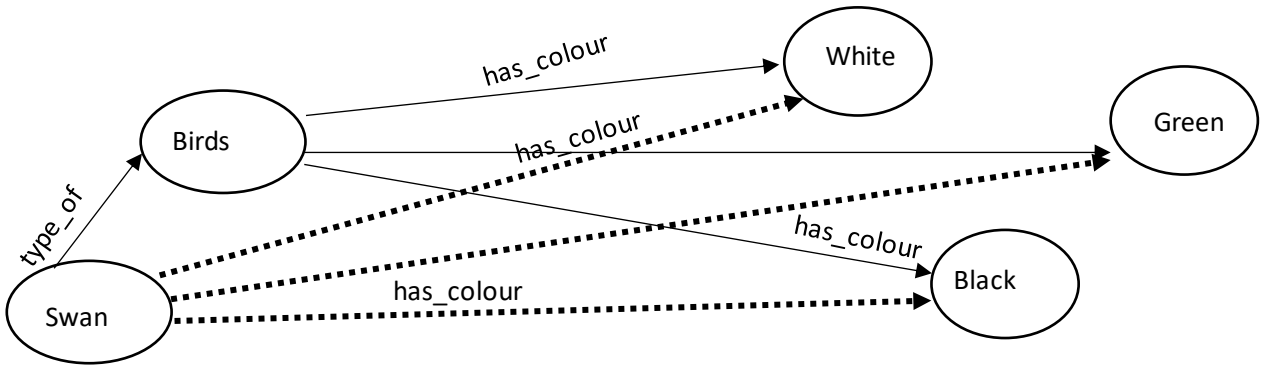

Figure 5. Knowledge Graph representing some facts about swans.

$$\exists Y(Swan, type\_of, Birds) \wedge (Birds, has\_colour, White) \rightarrow (Swan, has\_colour, White).$$

$$\exists Y(Swan, type\_of, Birds) \wedge (Birds, has\_colour, Black) \rightarrow (Swan, has\_colour, Black).$$

$$\exists Y(Swan, type\_of, Birds) \wedge (Birds, has\_colour, Green) \rightarrow (Swan, has\_colour, Green).$$

$$(Swan, has\_color, White) \wedge (Swan, has\_color, Black) \wedge (Swan, has\_color, Green) \quad (4)$$

Deducing the final relationship is logically correct, but such relation is not observed in the real-world, this inconsistency is actually beneficial, because in contrast with the large language model

where models could only stipulate that swans cannot be green, since there is very low learned positive correlation between the word *Green* and *Swan*, they cannot infer theoretical assumptions, about future observation, based on other relations that are actually not just statistically, but also logically connected. However, when we are dealing with semantic elicitation, we initially only care with what could be true in the present Knowledge Graph, not what could be true if future discovery is made (discovery of a green swan). To be able to trim relation we need to evaluate, how probable ones must be in a Knowledge Graph. In our approach we are going to first expand the Knowledge Graph on figure 4 with node features. By doing so, we are going to create computationally expensive expanded Knowledge Graph, however, we will apply one trick that could scale the graph for us. Let's first construct the node features. For the general porpoise of illustration, we construct a feature matrix which will be having the same dimensionality for each different node type. For nodes type colour we encode each colour as array of numbers RGB could be used green [0,128,0], white [255, 255, 255], black[128,128,128], then for *Swan* and *Birds*. We encode each basic colour with the inverse of the RGB code and set it as a column of dimensionality R^{NxM} where N-number of colours, M-number of encoded scheme, for RGB M = 3.

To keep the description brief we will consider only the three existing colours from the Knowledge Graph on figure 4, so the encoded feature matrix for *Swan* and *Birds* is R and for each node colour we sequentially C_1 for green, C_2 for black, C_3 for white

$$C_1 = \begin{pmatrix} 0 & 0 & 0 \\ 128 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad C_2 = \begin{pmatrix} 0 & 128 & 0 \\ 0 & 128 & 0 \\ 0 & 128 & 0 \end{pmatrix}, \quad C_3 = \begin{pmatrix} 0 & 0 & 256 \\ 0 & 0 & 256 \\ 0 & 0 & 256 \end{pmatrix},$$

$$\mathrm{R} = \begin{pmatrix} 0 & 128 & 0 \\ 128 & 128 & 128 \\ 256 & 256 & 256 \end{pmatrix} \tag{5}$$

Now we calculate a dot product between all of the matrixes C_i and R, we get the following matrixes, scaled by factor of 195509, since this is normalizing value for maximal value of 256 for an element[1]:

[1] In RGB scale the maximum value of an array element is 256, the maximum value that could be calculated by dot product is then 195509

$$C_1.\mathrm{R} = \begin{pmatrix} 0.08 & 0 & 0 \\ 0.08 & 0 & 0 \\ 0.2 & 0 & 0 \end{pmatrix},\ C_2.\mathrm{R} = \begin{pmatrix} 0 & 0.08 & 0 \\ 0 & 0.3 & 0 \\ 0 & 0.5 & 0 \end{pmatrix},\ C_3.\mathrm{R} = \begin{pmatrix} 0 & 0 & 0.2 \\ 0 & 0 & 0.5 \\ 0 & 0 & 1 \end{pmatrix} \quad (6)$$

For nodes type *Swan* we create matrix, $B = \begin{pmatrix} 0.08 & 0.08 & 0.2 \\ 0.08 & 0.3 & 0.5 \\ 0.2 & 0.5 & 1 \end{pmatrix}$, for node type *Swan*, since we do not have information for existing link *has_colour* (this is what we try to predict) between {*Swam,*

has_colour, Green}, the matrix B omits the first column for colour green $B = \begin{pmatrix} 0 & 0.08 & 0.2 \\ 0 & 0.3 & 0.5 \\ 0 & 0.5 & 1 \end{pmatrix}$.

We will call this matrix B, a *bias* matrix.

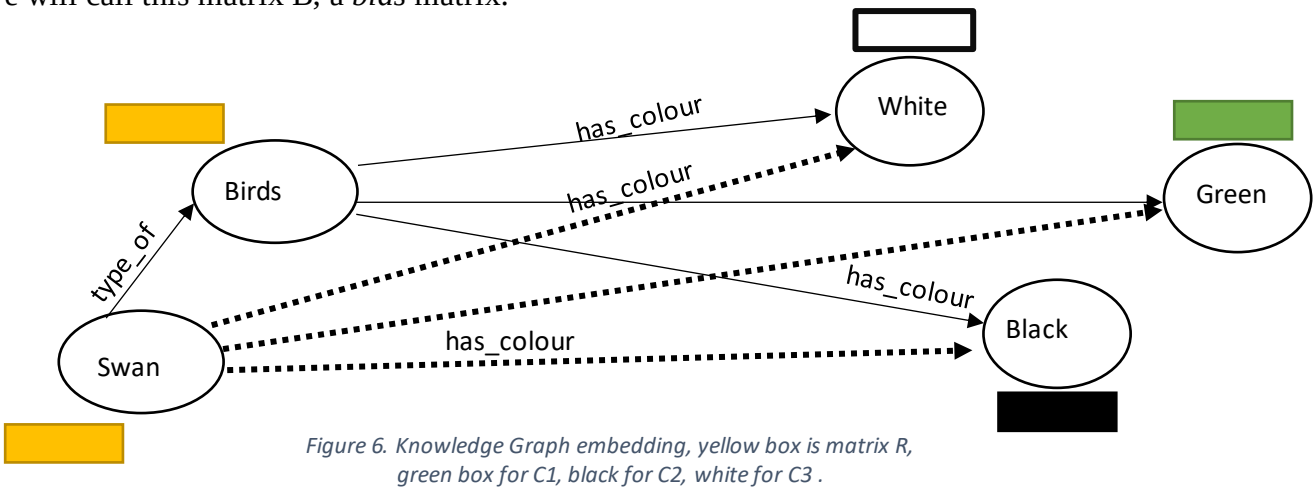

Figure 6. Knowledge Graph embedding, yellow box is matrix R, green box for C1, black for C2, white for C3 .

Our task is now to analyze the process of semantic relation discovery, from the perspective of machine learning algorithm. We will apply supervised mechanism to learn parameterized representation of a training graph, for the porpoise of the illustration we will take the graph from figure 4, but we will mark that the relations between *Swan* and *Black*, *White*, exist. To learn this parameterized representation, we will use a novel state-of-the-art algorithm called Graph Neural Network (GNN) and we will apply this algorithm to score the likelihood of a triplet $k = (s, P, o)$, we will refer to such triplet as a tuple, since in caries standard data annotation and is convenient

for the aims of the investigation, and we denote a tuple as *(u, r_t, v)*, given that $G_{(u, v, rt)}$ are the extracted and labeled subgraphs with tuples{2}, the overall task is to calculate what is the probability of a tuple *(u, r_t, v)*, to be predicted as having relationship between the *head* node *u* and *tail* node *v* in a Knowledge Graph, where we refer to *u,v* as target nodes and r_t as target relation. GNN adopts message-passing scheme{3}, where a node representation s iteratively updated by combining it with aggregation of its neighbors' representation. In particular the k^{th} layer of a GNN is given by,

$$a_t^k = AGGREGATE^k\left(h_s^{k-1}: s \in N(t), h_t^{k-1}\right), (7)$$

$$h_t^k = COMBINE^k\left(h_t^{k-1}, a_t^k\right), (8)$$

Where a_t^k is the aggregated message from the neighbors, h_t^k denotes the latent representation of node *t* in *k*-th layer and *N(t)* denotes the set of immediate outgoing neighbors of node *t*. Each node *i*, in the subgroup around *u* and *v* is labeled with the tuple *(d(i,u), d(i, v))*, where *d(i, u)* denotes the shortest distance between nodes *i* and *u*, without counting any path through *v*, This capture the topological position of each node with respect to the target nodes and reflects its structural role in the subgraph. The two target nodes *u,v* are uniquely labeled (0, 1) and (1, 0). The node features for the semantic rule discovery part are thus dot product between (0, 1) and (1, 0) vectors. In (7) The initial latent node representation of any node *i*, h_i^0, is initialized to the node features, X_i build according to the described scheme. We will also add attention layer to our calculation in accordance with [3] and complete the aggregation function in a full form,

$$a_t^k = \sum_{r=1}^{R} \sum_{s \in N_r(t)} \alpha_{rr_tst}^k \; W_r^k h_r^{k-1} + B_{t,r}^{k-1}, (9)$$

Where $\alpha_{rr_tst}^k$ could be any attention mechanism applicable for vertexes[4] W_r^k are the learnable parameters which the GNN must calculate for a training graph, and $B_{t,r,s}^{k-1}$ correspond to the matrix B that we calculated the notation *t,r,s* in B corresponds to the column between head node t colour relationship *r* and tail node s. By introducing these biases, we will induce higher score to be calculated for training over nodes with existing colour relations. The COMBINE function is given by

$$h_t^k = ReLU\ \left(W_{self}^k, h_t^{k-1}\ + a_t^k\ \right), (10)$$

With the GNN architecture as described above, we obtain the node representations after *L* layers of message passing. A subgraph representation of $G_{(u,\,v,\,rt\,)}$ is obtained by averaging of all the latent node representations

$$h_{\mathrm{G(u,v,}rt\,)}^L = \frac{1}{|V|}\sum_{i\,\epsilon\,V} h_i^L\ , (11)$$

Where *V* denotes the set of vertexes in graph $G_{(u,\,v,\,rt\,)}$. To obtain the score for the likelihood in a tuple *(u, r_t, v)*, we concatenate the subgraph representation $h_{\mathrm{G(u,v,}rt\,)}^L$, the target nodes' latent representation h_u^L and h_v^L, and a learned embedding of the target relation e_{r_t}, and pass these concatenations through linear layer,

$$score_{(u,rt,v)} = W^T\,[h_{\mathrm{G(u,v,}rt\,)}^L \oplus\ h_u^L \oplus\ h_v^L \oplus e_{r_t}]\,, (12)$$

We then use the following loss function to train our model via stochastic gradient descent:

$$\varphi = \sum_{i=1}^{|\varepsilon|} max(0, score(n_i) - score(p_i) + \gamma), (13)$$

where ε is the set of all vertexes/tuples in the training graph p_i and n_i denote the positively and negatively tuples respectively and γ is a hyperparameter.

At the end if we take again the example in (4), into it the tuples below would have much higher score due to the node embedding of matrix *B* that we have calculated (6), (7)

$$(Swan, has\_color, White) \wedge (Swan, has\_color, Black)$$

the rule in (4) can then be pruned to discard any relation that does not pass a numerical threshold for example $\delta\ >\ 0.5$, which will happen with the relation $(Swan, has\_color, Green)$, since the node embedding for the initial relation will have zero values in the *bias* matrix.

In conclusion in this paper, we developed two step approach for semantic elicitation and representation of relationships between node entities in graph. The method first use node embedding to calculate a *bias* matrix which induce weight to be calculated in the second step of rule creation with emphasize of connections that exist in the training graph, but do not block rules with lower score from creation at the end is a task of fine tuning a threshold parameter to be

introduced where relationships will be eventually pruned if threshold criteria is not met. Limitation of the proposed theoretical framework is the computational complexity which is introduced by searching several times through the graphs, for first step node embedding and then for capturing tuples and calculating scores in subgraphs. Although mathematical analysis points out that semantic analysis by this proposed framework has been completed, this paper does not elaborate an experimental part, which is reserved for future work.

Notes:

1. Node embedding is a technique, which represents graph in continuous vector space, there are numerous techniques. See *Watch Your Step: Learning Node Embeddings via Graph Attention*

2.. See *Inductive Relation Prediction by Subgraph Reasoning Zhang, M. and Chen, Y. Link prediction based on graph neural networks,* the paper contains exhaustive analyses on the technique.

3. The idea of message passing networks was introduced in a paper by Gilmer et al. in 2017 and it essentially boils GNN layers down to three main steps:

1. Every node in the graph computes a **message** for each of its neighbors. Messages are a function of the node, the neighbor, and the edge between them.
2. Messages are sent, and every node **aggregates** the messages it receives, using a permutation-invariant function (i.e., it doesn't matter in which order the messages are received). This function is usually a sum or an average, but it can be anything.
3. After receiving the messages, each node **updates** its attributes as a function of its current attributes and the aggregated messages.

4. *Attention* is a compositional technique which is not native to graph neural network all classes of neural networks could be invoked with *attention,* basically this mechanism impose constrain of what should be learned with higher *attention* from a network. See *Attention Is All You Need.*

Bibliography:

Abu-El-Haija, S., Perozzi, B., Al-Rfou, R., & Alemi, A. A. 2018. Watch your step: Learning node embeddings via graph attention // Advances in neural information processing systems, 31.

Bianchi, F. M., Grattarola, D., Livi, L., & Alippi, C. 2021. Graph neural networks with convolutional arma filters // IEEE transactions on pattern analysis and machine intelligence, 44(7), 3496-3507.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., & Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data // Advances in neural information processing systems, 26.

Borgwardt, K. M., & Kriegel, H. P. 2005, November. Shortest-path kernels on graphs // Fifth IEEE international conference on data mining (ICDM'05) (pp. 8-pp). IEEE.

Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. 2013. Spectral networks and locally connected networks on graphs // arXiv preprint arXiv:1312.6203.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. 2017. Attention is all you need // Advances in neural information processing systems, 30.