Joint Completion and Alignment of Multilingual Knowledge Graphs

BTP Report

by

Shubham Lohiya
(Roll No. 18D100020)

Supervisors:
Prof. Soumen Chakrabarti
Prof. Asim Tewari

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY
2022

Contents

Chapter 1

Introduction

1.1 Background

Humans are inherently good at storing, reasoning and interpreting knowledge, and this enables us to learn to perform tasks from historically accumulated knowledge. Today there are billions of documents on the web. For a very long time, it has been the aim to computer science and AI to formulate a way for machines to interpret this *digital* knowledge through knowledge representation and reasoning. Recently, Knowledge Graphs have drawn a lot of research attention as form of structured fact representation. Figure 1.1 is an example of a small knowledge graph taken from [5].

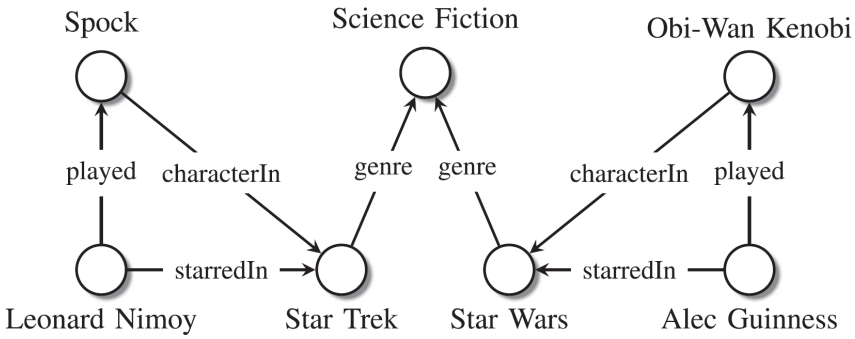

Figure 1.1: Sample knowledge graph. Nodes represent entities, edge labels represent types of relations, and edges represent existing relationships.

Knowledge graphs model information in the form of entities and relationships between them. A common form of fact representation in knowledge graphs is in the form of binary relationships, in particular `(subject, predicate, object)` or SPO triples, where the subject and the object are entities and the predicate is the relation between them. Entities can be real-world objects or abstract concepts. Another way to represent such fact triples is `(h, r, t)` which indicates a relation `r` between head entity `h` and tail entity `t`. In this report, the notation 'e' points to an

entity e, and '**e**' refers to the feature vector representation or embedding of e. Similar notation follows for relations.

Some examples of real-world knowledge graphs include Freebase [2], DBPedia [1], Yago [9] and WikiData [14]. Knowledge graphs are extensively used on tasks like search engines, chatbots, and recommendation systems. Biological Knowledge Graphs are also being researched and have applications like understanding molecular biology and drug discovery. However, knowledge graphs are generally far from exhaustive, and have especially sparse representation in low-resource languages. This has prompted research into techniques for Knowledge Graph Completion for predicting missing facts, and Knowledge Graph Alignments for integrating information from various graphs.

1.2 Report outline

The subject matter of the report is presented in the following 6 chapters,

- ✓ Chapter-1 provides an overview Knowledge Graphs (KGs) as a way of representing knowledge in the form of fact triples and discusses applications. It also introduces some problems with KGs that serve as the subject of recent research efforts. The outline of the report is also mentioned in this chapter.
- ✓ Chapter-2 describes all the new developments of methodologies for KG completion and alignment in separate subsections.
- ✓ Chapter 3 discusses the datasets and benchmarks for various knowledge graph research problems. It also presents important statistics and inferences gained from analyses of these datasets.
- ✓ Chapter-4 presents the AlignKGC Framework
- ✓ Chapter-5 explains the experiments conducted and highlights the findings.
- ✓ Chapter-6 concludes the report by summarizing the research conducted and obtained results. Future work as a continuation of this research work is also proposed.

Chapter 2

Literature Review

2.1 KG completion

Knowledge Graph Completion (KGC) is the task of inferring missing facts based on existing data in a knowledge graph. Described below are some recent notable approaches for KGC.

2.1.1 ComplEx

ComplEx [13] calculates the confidence score of a candidate triple (s,r,o) using a Hermitian dot product on complex vector representations.

$$Pr(s,r,o) = \sigma(\Re(\langle \mathbf{s}, \mathbf{r}, \bar{\mathbf{o}} \rangle)) = \sigma(\Re(\sum_{k=1}^{K} r_k s_k \bar{o}_k)) \tag{2.1}$$

where $\mathbf{s}$, $\mathbf{r}$, and $\mathbf{o} \in \mathbb{C}^K$, $\Re(\cdot)$ takes the real part of a complex number, and $\bar{\cdot}$ denotes complex conjugate. The scoring function is quite similar to DistMult [17] which used a multi-linear dot product of real vectors and was unable to model anti-symmetric relations. ComplEx also has the same time and space complexity as TransE [3], which was unable to model symmetric relations. ComplEx is can handle both symmetric and anti-symmetric relations.

2.1.2 RotatE

RotatE [11] is an approach for knowledge graph embedding which defines each relation as a rotation from the head entity to the tail entity in the complex vector space. It improves over ComplEx, and is capable of modelling composition relations unlike ComplEx. RotatE uses a distance based scoring function which is defined as:

$$\phi(s,r,o) = -\left\|(\mathbf{s} \circ \mathbf{r} - \mathbf{o})\right\|^2 \tag{2.2}$$

where ∘ denotes the Hadamard Product.

RotatE also introduces a new approach to negative sampling during optimization. It shows that self-adversarial negative sampling method which samples negative triples according to the current embedding model is more effective than uniform negative sampling.

2.2 Entity and relation alignment

Entity Alignment is the task of finding entities in two KGs that refer to the same real-world object. It plays a vital role in automatically integrating multiple knowledge bases. Relations Alignment similarly aims to align relations in two KGs that refer to the same underlying relation. Described below are some recent approaches for the entity and relation alignment tasks.

2.2.1 BERT-INT

BERT-based Interaction Model For Knowledge Graph Alignment [12] is currently one of the best performing models for entity alignment. BERT-INT posits that due to heterogenity of different knowledge graphs and low available seed alignment information, alignment methods that primarily rely on graph structures to incorporate side information (such as name, description and attributes) result in noisy and inefficient results. It aims to leverage only side information and introduces an interaction module to capture fine-grained matches between neighbors and attributes. The BERT-INT architecture (refer Figure 2.1) consists of two modules:

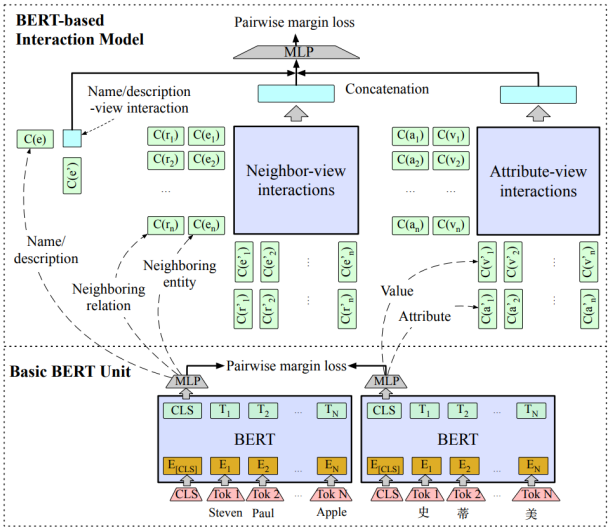

Figure 2.1: The BERT-INT framework for entity alignment.

1. **Basic BERT Unit**

This module uses pre-trained multilingual BERT[1] embeddings to calculate closest matches. Training data is of the form $\mathscr{D} = (e, e'^{+}, e'^{-})$. For each entity e in the dataset, the basic BERT unit takes its name/description as input to obtain $C(e) = MLP(CLS(e))$. The BERT model is fine-tuned with a pairwise margin loss seen in Equation 2.3.

$$\mathscr{L} = \sum_{(e,e'^{+},e'^{-})\in\mathscr{D}} max\{0, g(e,e'^{+}) - g(e,e'^{-}) + m\} \tag{2.3}$$

where m is the margin enforced between positive and negative pairs, and $g(e,e')$ is the l_1 distance capturing similarity between $C(e)$ and $C(e')$.

2. **BERT-based Interaction Model**

The interaction model is based on BERT and consists of the following:

(a) **Name/description-view interaction**: The cosine similarity between $C(e)$ and $C(e')$ is calculated as the name/description-view interaction.

(b) **Neighbor-view Interactions**: The basic BERT unit is applied to obtain $C(e_i)_{i=1}^{|\mathscr{N}(e)|}$ and $C(e'_i)_{i=1}^{|\mathscr{N}(e')|}$ for e and e''s neighboring entities. Then, a similarity matrix is computed between the two embedding sets and a dual aggregation function is applied to extract the similarity features from the matrix. The matrix $\mathbf{S}$ represents the neighbor interaction based on cosine similarity. A a row-based similarity embedding $\phi^r(\mathscr{N}(e),\mathscr{N}(e'))$ is calculated as described in equation 2.4.

$$\begin{aligned} s_i^{max} &= \max_{j=0}^{n}\{s_{i0},\cdots,s_{ij},\cdots,s_{in}\} \\ K_l(s_i^{max}) &= \exp\left[-\frac{(s_i^{max}-\mu_l)^2}{2\sigma_l^2}\right] \\ \mathbf{K}^r(\mathbf{S}_i) &= [K_1(s_i^{max}),\cdots,K_l(s_i^{max}),\cdots,K_L(s_i^{max})] \\ \phi^r(\mathscr{N}(e),\mathscr{N}(e')) &= \frac{1}{|\mathscr{N}(e)|}\sum_{i=1}^{|\mathscr{N}(e)|}\log\mathbf{K}^r(\mathbf{S}_i) \end{aligned} \tag{2.4}$$

A a column-based similarity embedding $\phi^c(\mathscr{N}(e),\mathscr{N}(e'))$ is computed in a similar manner. The final similarity embedding is computed by concatenating them.

$$\phi(\mathscr{N}(e),\mathscr{N}(e')) = \phi^r(\mathscr{N}(e),\mathscr{N}(e')) \oplus \phi^c(\mathscr{N}(e),\mathscr{N}(e')) \tag{2.5}$$

[1] https://github.com/google-research/bert

(c) **Neighboring Relation Mask Matrix**: The similarity matrix **S** is modified using matrix **M** which calculated from relation embeddings $C(r)$. $C(r)$ is computed by concatenating averaged embeddings of all head and tail entities associated with relation r. Finally **S** is recalculated as follows:

$$S_{ij} = S_{ij} \otimes M_{ij} \tag{2.6}$$

(d) **Interactions between Multi-hop Neighbors**: In the similarity matrix instead of taking 1-hop neighbours, m-hop neighbours are used.

(e) **Attribute-view Interactions**: $\phi(\mathscr{A}(e), \mathscr{A}(e'))$ are computed in the same manner as neighborhood interaction

The final interaction embedding is computed as equation 2.7 and the similarity score is defined in equation 2.8.

$$\phi(e,e') = [\phi(\mathscr{N}(e), \mathscr{N}(e')) \oplus \phi(\mathscr{A}(e), \mathscr{A}(e')) \oplus cos(C(e), C(e'))] \tag{2.7}$$

$$g(e,e') = MLP(\phi(e,e')) \tag{2.8}$$

2.2.2 RNM

Most methods for EA aggregate information from neighboring nodes but they ignore the relations between entities, which are also important for neighborhood matching. The Relation-aware Neighborhood Matching model for entity alignment (RNM) [21] pays attention to the positive interactions between the entity alignment and relation alignment. It presents an iterative framework designed to leverage the positive interactions between the EA and RA in a semi-supervised manner. The components of the proposed architecture (refer Figure 2.2) are described below:

1. **Preliminaries**: The objective is to align entities given two heterogeneous KGs, $G_1 = (E_1, R_1, T_1)$ and $G_2 = (E_1, R_1, T_1)$. A set $\mathbb{L} = \{(e_1, e_2) | e_1 \in E_1, e_2 \in E_2, e_1 \; equals \; to \; e_2\}$ is provided to the model as seed alignment.

2. **Embedding Learning for Entity and Relation**:

   Entity Embedding: GCNs are utilized to embed all entities of the two KGs into the same latent space. Pre-trained word embeddings are used to initialize the entity representations.

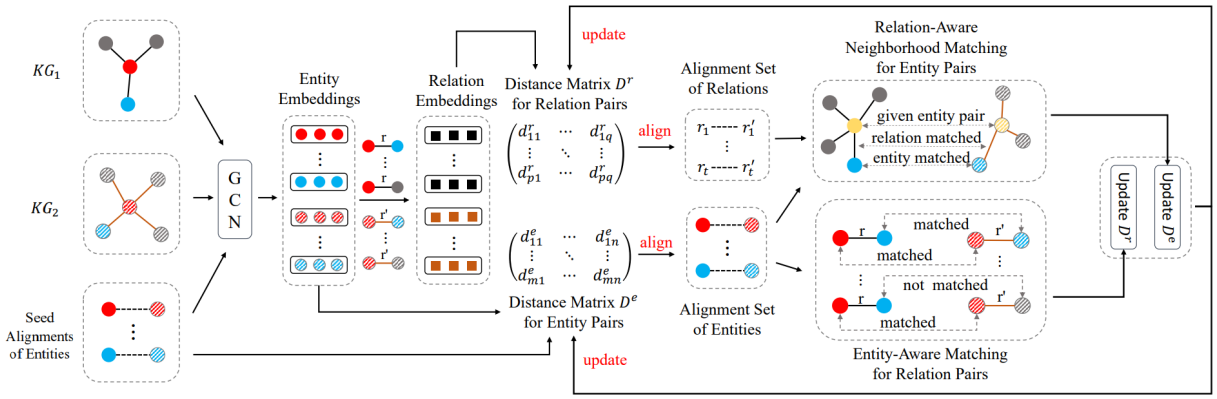

Figure 2.2: Overall architecture of the RNM model for entity and relation alignment.

The outputs of the GCN define the final entity representations as $\tilde{\mathbf{X}} = \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \cdots \tilde{\mathbf{x}}_n | \tilde{\mathbf{x}}_i \in \mathbb{R}^{\tilde{d}}\}$. For an entity pair (e_i, e'_j), the distance between them is defined as

$$d(e_i, e'_j) = ||\tilde{\mathbf{x}}_{\mathbf{e_i}} - \tilde{\mathbf{x}}_{\mathbf{e'_j}}||_1 \tag{2.9}$$

The embeddings are trained by optimizing the margin-based loss given in equation 2.10

$$L_E = \sum_{(p,q)\in\mathbb{L}} \sum_{(p',q')\in\mathbb{L}'} \max\left\{0, d(p,q) - d\left(p',q'\right) + \gamma\right\} \tag{2.10}$$

where $\mathbb{L}'$ is a set of negative alignments created using nearest neighbor sampling, and $\gamma > 0$ denotes the margin.

Relation Embedding: The relation embedding is computed as follows

$$\boldsymbol{r} = \text{concat}\left[\boldsymbol{g}_r^h, \boldsymbol{g}_r^t\right] \tag{2.11}$$

where $\boldsymbol{r} \in \mathbb{R}^{2\tilde{d}}$, and $\boldsymbol{g}_r^h$ and $\boldsymbol{g}_r^t$ denote average embeddings of all distinct head and tail entities for r.

To further explore the translational information for relations based on triples, a regularizer is defined as

$$\Omega_R = \sum_{(h,r,t)\in T_1 \cup T_2} \|\boldsymbol{h} + \boldsymbol{W}_R\boldsymbol{r} - \boldsymbol{t}\|_1 \tag{2.12}$$

where $\boldsymbol{W}_R \in \mathbb{R}^{\tilde{d}\times 2\tilde{d}}$ denotes the transformation matrix from the latent relation space to the latent entity space that has to be learned.

The joint learning objective is defined in equation 2.13

$$L = L_E + \lambda \cdot \Omega_R \tag{2.13}$$

3. **Relation-Aware Neighborhood Matching**: GCNs aim to aggregate information from neighboring nodes but may also bring some additional noise from neighbors. To reduce the impact of these noise, Relation-Aware neighborhood matching is done and the entity distance matrix is iteratively updated. If two entities from different KGs are equivalent, then with an equivalent relation, the alignment probability of two pointing tail entities is inferred to be 1 for a 1-to-1 relation while 1-to-N relation can only show the probability of $1/N$.

   For neighborhood matching with respect to e_i and e'_j, all the entity pairs and the connected relation pairs in $C^e_{ij} = \{(n_1,n_2),(r_1,r_2) \mid n_1 \in \mathscr{N}_{e_i}, n_2 \in \mathscr{N}_{e'_j}, (e_i,r_1,n_1) \in T_1, (e'_j,r_2,n_2) \in T_2\}$ are to be compared. Then the focus is on the matched neighbors with matched relations which are vital for entity alignment, so we define the matched set M^e_{ij} as the subset of C^e_{ij}, in which the elements satisfy $(n_1,n_2) \in \mathbb{L}_e$ and $(r_1,r_2) \in \mathbb{L}_r$, where $\mathbb{L}_e$ denotes the alignment set of entities and $\mathbb{L}_r$ denotes the alignment set of relations. These sets are obtained every iteration using the seed alignments and additional discovered alignments by thresholding the distance matrices.

   For each matched case in M^e_{ij}, the alignment probability is computed as follows,

$$\begin{aligned} P(r_1,r_2,n_1,n_2) &= P(r_1,n_1) \cdot P(r_2,n_2) \\ \text{where} \quad P(r_1,n_1) &= \frac{1}{|\{e \mid (e,r_1,n_1) \in T_1\}|} \\ \text{and} \quad P(r_2,n_2) &= \frac{1}{|\{e \mid (e,r_2,n_2) \in T_2\}|} \end{aligned} \tag{2.14}$$

   and the distance between two entities is updated as follows,

$$d^e_{ij} = ||\tilde{\mathbf{x}}_{e_i} - \tilde{\mathbf{x}}_{e'_j}||_1 - \lambda_e \cdot \frac{\sum_{M^e_{ij}} P(r_1,r_2,n_1,n_2)}{|\mathscr{N}_{e_i}| + |\mathscr{N}_{e'_j}|} \tag{2.15}$$

   where λ_e is a hyper-parameter to control the trade-off between the embedding distance and the matching score. Greater matching score indicates the higher probability of alignment for the candidate entity pair.

4. **Entity-Aware Relation Matching**:

   For two relations from different KGs, it is assumed that high number of (head, tail) alignments of the corresponding triples accross the KGs indicates higher likelihood that the two relations are equivalent. $S_r = \{(h,t) \mid (h,r,t) \in T\}$ is defined as the set of its

related entity pairs. Given a candidate relation pair (r_i, r'_j), all entity pairs in $C^r_{ij} = \{(h_1,h_2),(t_1,t_2) \mid (h_1,t_1) \in S_{r_i}, (h_2,t_2) \in S_{r'_j}\}$ are to be compared and the matching set M^r_{ij} is defined as the subset of C^r_{ij} where elements meet the conditions of $(h_1,h_2) \in \mathbb{L}_e$ and $(t_1,t_2) \in \mathbb{L}_e$. Then, the distance between the two relations is updated as follows,

$$d^r_{ij} = \|\boldsymbol{r}_i - \boldsymbol{r}'_j\|_1 - \lambda_r \cdot \frac{|M^r_{ij}|}{|S_{r_i}| + |S_{r'_j}|} \tag{2.16}$$

where λ_r is the trade-off coefficient.

2.2.3 RAGA

There are still two critical challenges for entity alignment that are not sufficiently addressed by existing approaches:

1. Most TransE-based methods regard relations as the translation between two entities. These methods are limited by the uniqueness of the relation between two entities. Thus the first challenge to EA is how to represent entities in a manner that sufficiently utilizes multiple relations

2. Local alignment techniques often leads to conflicts in birectional alignment. For eg. an entity may be the common best match for several entities; or an entity's best match might have some other entity as its best match. Thus the second challenge is how to align entities of two KGs from a global perspective

RAGA or Relation-aware Graph Attention Networks for Global Entity Alignment [20] attempts to address the above two challenges by (1) capturing interactions between entities and relations, which leads to sufficient utilization of multiple relations between entities, and by (2) designing a global alignment algorithm based on deferred acceptance algorithm and a more fine-grained similarity matrix.
The RAGA framework consists of four parts as seen in Figure 2.3 which are described below:

1. **Basic Neighbor Aggregation Networks:** GCNs are used to explicitly encode entities in KGs with structure information. The ouput of the l-th GCN layer is computed as:

$$\boldsymbol{X}^{(l+1)} = \text{ReLU}\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}\boldsymbol{X}^{(l)}\right), \tag{2.17}$$

where $\boldsymbol{X}$ is the entity embeddings matrix, $\tilde{A} = A + I, A$ is the adjacency matrix of KG, I is an identity matrix, and $\tilde{D}$ is the diagonal node degree matrix of $\tilde{A}$.

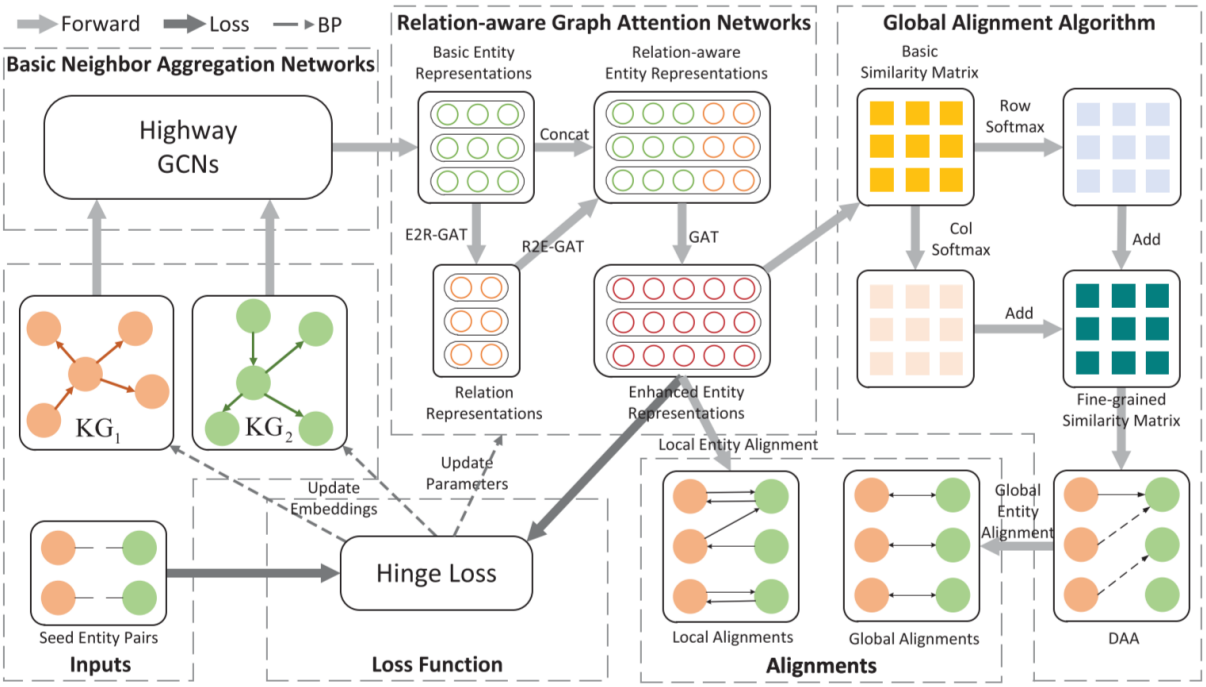

Figure 2.3: Overall architecture of the RAGA Framework.

Inspired by RDGCN [16], RAGA employs layer-wise Highway Networks to control the balance of the information between the entity itself and neighbour entities. The output of a Highway Network layer is the weighted sum of its input and the original output via gating weights:

$$\begin{aligned} T\left(\boldsymbol{X}^{(l)}\right) &= \sigma\left(\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)} + \boldsymbol{b}^{(l)}\right) \\ \boldsymbol{X}^{(l+1)} &= T\left(\boldsymbol{X}^{(l)}\right) \cdot \boldsymbol{X}^{(l+1)} + \left(1 - T\left(\boldsymbol{X}^{(l)}\right)\right) \cdot \boldsymbol{X}^{(l)} \end{aligned} \tag{2.18}$$

where σ is a sigmoid function, $\cdot$ is element-wise multiplication, $\boldsymbol{W}^{(l)}$ and $\boldsymbol{b}^{(l)}$ are the weight matrix and bias vector for the transform gate of the l-th layer.

2. **Relation-aware Graph Attention Networks:** Entity representations are passed through three diffusion modes of entity to relation, relation to entity, and entity to entity to obtain more accurate entity representations.

   Relation Representations: Relations should have a different amount of information and different information space from entity representations as the distribution of relations is denser than the distribution of entities. For relation r_k, the head entity representation $\boldsymbol{r}_k^h$ is computed as follows:

$$\alpha_{ijk} = \frac{\exp\left(\text{LeakReLU}\left(\boldsymbol{a}^T\left[\boldsymbol{x}_i\boldsymbol{W}^h \| \boldsymbol{x}_j\boldsymbol{W}^t\right]\right)\right)}{\sum_{e_{i'}\in\mathscr{H}_{r_k}}\sum_{e_{j'}\in\mathscr{T}_{e_i r_k}}\exp\left(\text{LeakReLU}\left(\boldsymbol{a}^T\left[\boldsymbol{x}_{i'}\boldsymbol{W}^h \| \boldsymbol{x}_{j'}\boldsymbol{W}^t\right]\right)\right)},$$
$$\boldsymbol{r}_k^h = \text{ReLU}\left(\sum_{e_i\in\mathscr{H}_{r_k}}\sum_{e_j\in\mathscr{T}_{e_i r_k}}\alpha_{ijk}\boldsymbol{x}_i\boldsymbol{W}^h\right) \tag{2.19}$$

where α_{ijk} represents attention weight from head entity e_i to relation r_k based on head entity e_i and tail entity e_j, $\mathscr{H}_{r_k}$ is the set of head entities for relation r_k, $\mathscr{T}_{e_i r_k}$ is the set of tail entities for head entity e_i and relation r_k, $\boldsymbol{a}$ is a one-dimensional vector to map the $2d_r$-dimensional input into a scalar, d_r is half of the dimension of relation embeddings, and $\boldsymbol{W}^h, \boldsymbol{W}^t \in \mathbb{R}^{d_e \times d_r}$ are linear transition matrices for head and tail entity representation of relations respectively.

The tail entity representation $\boldsymbol{r}_k^t$ can be computed through a similar process, and then both are added to obtain the relation representation $\boldsymbol{r}_k$: $\boldsymbol{r}_k = \boldsymbol{r}_k^h + \boldsymbol{r}_k^t$.

Relation-aware Entity Representations: For entity e_i, an attention mechanism is used to calculate its out-relation embedding $\boldsymbol{x}_i^h$ and in-relation embedding $\boldsymbol{x}_i^t$ separately. $\boldsymbol{x}_i^h$ is computed as follows:

$$\alpha_{ik} = \frac{\exp\left(\text{LeakReLU}\left(\boldsymbol{a}^T\left[\boldsymbol{x}_i \| \boldsymbol{r}_k\right]\right)\right)}{\sum_{e_j\in\mathscr{T}_{e_i}}\sum_{r_{k'}\in\mathscr{R}_{e_i\varepsilon_j}}\exp\left(\text{LeakReLU}\left(\boldsymbol{a}^T\left[\boldsymbol{x}_i \| \boldsymbol{r}_{k'}\right]\right)\right)}$$
$$\boldsymbol{x}_i^h = \text{ReLU}\left(\sum_{e_j\in\mathscr{T}_{e_i}}\sum_{r_k\in\mathscr{R}_{e_i e_j}}\alpha_{ik}\boldsymbol{r}_k\right) \tag{2.20}$$

where α_{ik} represents attention weight from relation r_k to entity e_i, $\mathscr{T}_{e_i}$ is the set of tail entities for head entity e_i and $\mathscr{R}_{e_i e_j}$ is the set of relations between head entity e_i and tail entity e_j.

Then the relation-aware entity representations $\boldsymbol{x}_i^{rel}$ are obtained by concatenating $\boldsymbol{x}_i, \boldsymbol{x}_i^h$ and $\boldsymbol{x}_i^t$:

$$\boldsymbol{x}_i^{rel} = \left[\boldsymbol{x}_i \left\| \boldsymbol{x}_i^h \right\| \boldsymbol{x}_i^t\right] \tag{2.21}$$

Enhanced Entity Representations: One layer of ordinary graph attention networks is applied to enhance the influence of relations on two-hop entities. For entity e_i, the final output of embedding $\boldsymbol{x}_i^{\text{out}}$ can be computed by:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\boldsymbol{a}^T \left[\boldsymbol{x}_i^{rel} \| \boldsymbol{x}_j^{rel}\right]\right)\right)}{\sum_{j' \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(\boldsymbol{a}^T \left[\boldsymbol{x}_i^{rel} \| \boldsymbol{x}_{j'}^{rel}\right]\right)\right)}, \qquad \boldsymbol{x}_i^{\text{out}} = \left[\boldsymbol{x}_i^{rel} \| \text{ReLU}\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \boldsymbol{x}_i^{rel}\right)\right] \tag{2.22}$$

3. **End-to-End Training:** The similarity of entities is calculated using the Manhattan distance: $\text{dis}\left(e_i, e_j\right) = \left\| \boldsymbol{x}_i^{\text{out}} - \boldsymbol{x}_j^{\text{out}} \right\|_1$.

   Hinge Loss is used as the loss function for training:

$$L = \sum_{\left(e_i, e_j\right) \in T} \sum_{\left(e_i', e_j'\right) \in T'_{\left(e_i, e_j\right)}} \max\left(\text{dis}\left(e_i, e_j\right) - \text{dis}\left(e_i', e_j'\right) + \lambda, 0\right) \tag{2.23}$$

   where $T'_{\left(e_i, e_j\right)}$ is the set of negative sample for e_i and e_j and λ is the margin.

4. **Global Alignment Algorithm:** As optimal local matches for entity alignment may lead to many-to-one alignments that reduce performance and bring ambiguity to entity alignment results, entities should be aligned globally. A similarity matrix $S \in \mathbb{R}^{|E_1| \times |E_2|}$ can be constructed based on the Manhattan distance between every entity pair between the two KGs. According to prior knowledge, entity alignment is a bidirectional match problem between two KGs. Thus, a fine-grained similarity matrix S^g is calculated by applying softmax on both rows and columns of S and adding them together to get the fine-grained similarity matrix S^g :

$$S_{i,j}^g = \frac{\exp\left(S_{i,j}\right)}{\sum_{j'=1}^{|E_2|} \exp\left(S_{i,j'}\right)} + \frac{\exp\left(S_{i,j}\right)}{\sum_{i'=1}^{|E_1|} \exp\left(S_{i',j}\right)} \tag{2.24}$$

   Finally, DAA [7] is applied to the fine-grained similarity matrix S^g to get global alignments.

2.3 Text and attribute features

Text descriptions of entities and attributes can provide useful context signals for KG completion and alignment tasks. Described next are some techniques to incorporate this information for such tasks.

2.3.1 mBERT embeddings

BERT is a language representation model that can be used to obtain general purpose contextual word embeddings. BERT and its variants have been highly successful on various downstream NLP tasks, achieving state of the art results. Unlike fixed word embeddings like Word2Vec and GloVe, BERT embeddings capture context from surrounding text. mBERT is a Multilingual version of BERT.

BERT takes as input, N wordpiece tokens: $(x_1, \ldots, x_N)$. Following that, L layers of D-dimensional contextual representations $\mathbf{H}_i \in \mathbb{R}^{N \times D}$ are calculated by successive application of non-linear functions $\mathbf{H}_i = \mathrm{F}_i(\mathbf{H}_{i-1})$. The non linear function, F_i, is a multi-headed self-attention layer followed by a position-wise multi-layer perceptron (MLP).

$$\mathrm{F}_i(\mathbf{H}_{i-1}) = \mathrm{TransformerBlock}(\mathbf{H}_{i-1}) = \mathrm{MLP}(\mathrm{MultiHeadAttn}(\mathbf{H}_{i-1}, \mathbf{H}_{i-1}, \mathbf{H}_{i-1})) \quad (2.25)$$

BERT-INT [12] demonstrated the benefits of using text embeddings for entity alignment, by achieving state of the art results using just this side information instead of graph structures. It used mBERT embeddings to perform name/description-view interaction, neighbor-view interaction with relation masking, and attribute-view interaction as described in section 2.2.1. KG-BERT [18] has also demonstrated success in knowledge graph completion.

2.3.2 GloVe

GloVe [6] is an unsupervised learning algorithm for obtaining vector representations for words. As previously shown, the RNM model for entity alignment uses GloVe representations to initialise their GCN embeddings. Entity names from different languages are first translated to english to obtain the GloVe representations. As described in Chapter 5, experiments with RNM on DBP-5L and DBP15K datasets using both mBERT and GloVe initialisations yield some interesting insights about the benefit of using these vectors.

2.3.3 Other methods to generate entity representations

This sections describes a few methods to generate set representations of entities as mentioned in [15]. The paper used these set representations to perform Locality-Sensitive Hashing (LSH) for obtaining candidate pairs for entity alignment.

- **N-grams of Names**: If entity names are available and in the same language, this method generates a set of character-level n-grams of entities' names as the set-representations of entities.
- **N-grams of Attributes**: This method treats attribute values of an entity as text strings, and generates character-level n-grams of all the attribute values for each entity. All the n-grams are then merged into a set as the representation of the entity.
- **Seeding alignments**: If seeding alignments between two KGs are available, a set of aligned entities in an entity's neighborhood will be taken as the set-representation.

Chapter 3

Datasets

3.1 DBP-5L

DBP-5L contains KGs for 5 languages - Greek, Japanese, Spanish, French and English. It was sampled from the DBPedia knowledge base. This dataset is applicable to all three tasks – KGC, EA and RA

3.1.1 Data Statistics

Salient statistics of the DBP-5L dataset are mentioned in table 3.1

Language	Greek	Japanese	Spanish	French	English
#Entity	5,231	11,805	12,382	13,176	13,996
#Relation	111	128	144	178	831
#Triples	13,839	28,774	54,066	49,015	80,167

Table 3.1: Data distribution statistics for DBP5L

For KGC, the split of fact triples between the training, the validation and the test sets is approximately in the ratio of *60:30:10* for all 5 languages. Similar splits have been followed for sampling while converted other datasets like DBP15K to allow them to be used for testing KGC.

3.2 DBP15K

DBP15k is the most popular benchmark for Entity Alignment, and was also sampled from the DBPedia knowledge base. The data set has three multilingual KG pairs: *ZH_EN* (Chinese-English), *JA_EN* (Japanese-English), and *FR_EN* (French English).

3.2.1 Data Statistics

Salient statistics of the datasets are mentioned in table 3.2

Datasets		Entities	Relationships	Attributes	Rel. Triples	Attr. triples
$DBP15K_{ZH-EN}$	Chinese	66,469	2,830	8,113	153,929	379,684
	English	98,125	2,317	7,173	237,674	567,755
$DBP15K_{JA-EN}$	Japanese	65,744	2,043	5,882	164,373	354,619
	English	95,680	2,096	6,066	233,319	497,230
$DBP15K_{FR-EN}$	French	66,858	1,379	4,547	192,191	528,665
	English	105,889	2,209	6,422	278,590	576,543

Table 3.2: Data distribution statistics for DBP15k

3.2.2 EA Results

Results of various approaches to entity alignment on the DBP15K benchmark are shown in table 3.3

Models	ZH-EN			JA-EN			FR-EN		
	Hits@1	Hits@10	MRR	Hits@1	Hits@ 10	MRR	Hits@1	Hits@10	MRR
MTransE (Chen et al. 2017)	30.8	61.4	0.364	27.9	57.5	0.349	24.4	55.6	0.335
IPTransE (Zhu et al. 2017)	40.6	73.5	0.516	36.7	69.3	0.474	33.3	68.5	0.451
BootEA (Sun et al. 2018)	62.9	84.8	0.703	62.2	85.4	0.701	65.3	87.4	0.731
AKE (Lin et al. 2019)	32.5	70.3	0.449	25.9	66.3	0.390	28.7	68.1	0.416
SEA (Pei et al. 2019)	42.4	79.6	0.548	38.5	78.3	0.518	40.0	79.7	0.533
GCN-Align (Wang et al. 2018)	41.3	74.4	0.549	39.9	74.5	0.546	37.3	74.5	0.532
KECG (Li et al. 2019)	47.8	83.5	0.598	49.0	84.4	0.610	48.6	85.1	0.610
MuGNN (Cao et al. 2019a)	49.4	84.4	0.611	50.1	85.7	0.621	49.5	87.0	0.621
NAEA (Zhu et al. 2019)	65.0	86.7	0.720	64.1	87.3	0.718	67.3	89.4	0.752
AliNet (Sun et al. 2020)	53.9	82.6	0.628	54.9	83.1	0.645	55.2	85.2	0.657
GMNN (Xu et al. 2019)	67.9	78.5	0.694	74.0	87.2	0.789	89.4	95.2	0.913
RDGCN (Wu et al. 2019a)	70.8	84.6	0.746	76.7	89.5	0.812	88.6	95.7	0.911
HGCN (Wu et al. 2019b)	72.0	85.7	0.768	76.6	89.7	0.813	89.2	96.1	0.917
NMN (Wu et al. 2020)	73.3	86.9	0.781	78.5	91.2	0.827	90.2	96.7	0.924
BERT-INT	**96.8**	**99**	**0.977**	**96.4**	**99.1**	**0.975**	**99.2**	**99.8**	**0.995**
RNM	*84.0*	*91.9*	*0.870*	*87.2*	*94.4*	*0.899*	*93.8*	*98.1*	*0.954*

Table 3.3: EA results on DBP15K. These have been taken from the RNM paper and BERT-INT results have been taken from the BERT-INT paper

This clearly shows that RNM and BERT-INT hugely outperform the previous methods for entity alignment. RNM makes use of both word embeddings and graph structures, whereas BERT-INT relies only on side information for entity alignment as discussed in section 2.2.1. Hence, these two are our primary algorithms of interest for comparison with AlignKGC.

3.3 OpenEA

OpenEA is the newest standard dataset for Entity Alignment. It uses three well-known KGs as its sources: DBpedia, Wikidata, and YAGO3. The dataset also includes two cross-lingual versions of DBpedia: English–French and English–German.

3.3.1 Data Statistics

There are two versions of datasets for each pair of KGs to be aligned. V1 is generated by directly using the IDS algorithm. V2 is generated by first randomly deleting entities with low degrees (d <= 5) in the source KG to make the average degree doubled, and then executing IDS to fit the new KG. The statistics of the datasets are shown in Figure 3.1.

Datasets	KGs	15K (V1)				15K (V2)				100K (V1)				100K (V2)			
		#Rel.	#Att.	#Rel tr.	#Att tr.	#Rel.	#Att.	#Rel tr.	#Att tr.	#Rel.	#Att.	#Rel tr.	#Att tr.	#Rel.	#Att.	#Rel tr.	#Att tr.
EN-FR	EN	267	308	47,334	73,121	193	189	96,318	66,899	400	466	309,607	497,729	379	364	649,902	503,922
	FR	210	404	40,864	67,167	166	221	80,112	68,779	300	519	258,285	426,672	287	468	561,391	431,379
EN-DE	EN	215	286	47,676	83,755	169	171	84,867	81,988	381	451	335,359	552,750	323	326	622,588	560,247
	DE	131	194	50,419	156,150	96	116	92,632	186,335	196	252	336,240	716,615	170	189	629,395	793,710
D-W	DB	248	342	38,265	68,258	167	175	73,983	66,813	413	493	293,990	451,011	318	328	616,457	467,103
	WD	169	649	42,746	138,246	121	457	83,365	175,686	261	874	251,708	687,860	239	760	588,203	878,219
D-Y	DB	165	257	30,291	71,716	72	90	68,063	65,100	287	379	294,188	523,062	230	277	576,547	547,026
	YG	28	35	26,638	132,114	21	20	60,970	131,151	32	38	400,518	749,787	31	36	865,265	855,161

Figure 3.1: Data Statistics for OpenEA

3.3.2 EA Results

Results of various approaches to entity alignment on the OpenEA benchmark are shown in Figure 3.2.

	EN_DE				EN_FR				DB_WIKI			
	15k		100k		15k		100k		15k		100k	
	v1	v2	v1	v2	v1	v2	v1	v2	v1	v2	v1	v2
MtransE	0.307	0.193	0.14	0.115	0.247	0.24	0.138	0.09	0.259	0.271	0.21	0.148
BootEA	0.675	0.833	0.518	0.739	0.507	0.66	0.389	0.64	0.572	0.821	0.516	0.766
JAPE	0.288	0.167	0.152	0.11	0.262	0.292	0.165	0.125	0.25	0.262	0.211	0.154
MultiKE	0.756	0.755	0.668	0.661	0.749	0.864	0.629	0.642	0.411	0.495	0.29	0.319
RDGCN	0.83	0.83	0.722	0.766	0.755	0.847	0.64	0.715	0.515	0.623	0.362	0.421
BERTINT	**0.98**	**0.96**	**0.97**	**0.98**	**0.983**	**0.99**	**0.97**	**0.975**	**0.98**	**0.96**	**0.97**	**0.98**

Figure 3.2: EA results on OpenEA taken from OpenEA's parent paper. BERT-INT results have been obtained by running their code.

3.4 DBP2.0

Since KGs possess different sets of entities, there could be entities that cannot find alignment across them, leading to the problem of *dangling entities*. As previous datasets do not contain dangling entities, Sun et al. [10] design a new dataset that can be used for both entity alignment and dangling entity detection. This Dataset was also sampled from DBPedia.

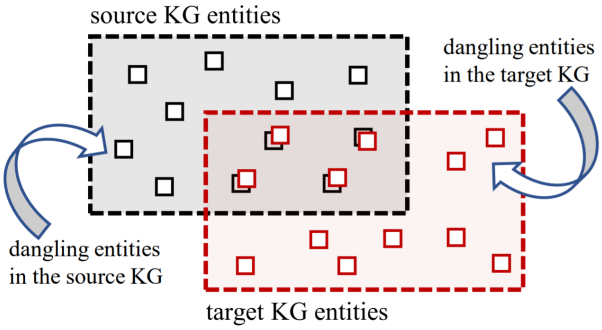

Figure 3.3: Illustration of entity alignment between two KGs with dangling cases. Paired red and black squares in the overlap region denote entity alignment while others are dangling entities without counterparts

3.4.1 Dataset Construction

A two-step dataset extraction is followed to ensure that the selected dangling entities are indeed without counterparts. First, two subgraphs are sampled for which all entities have alignments in the other subgraph. Then, a randomly selected disjoint set of entities are removed from the source and target graphs, to make their counterparts dangling.

3.4.2 Data Statistics

Salient statistics of the DBP2.0 dataset are mentioned in table 3.4

Datasets		# Entities	# Rel.	# Triples	# Align.
ZH-EN	ZH	84,996	3,706	286,067	33,183
	EN	118,996	3,402	586,868	
JA-EN	JA	100,860	3,243	347,204	39,770
	EN	139,304	3,396	668,341	
FR-EN	FR	221,327	2,841	802,678	123,952
	EN	278,411	4,598	1,287,231	

Table 3.4: Statistics of the DBP2.0 dataset

Chapter 4

AlignKGC

4.1 Introduction

The key objective of AlignKGC [8] is to recognize and exploit the synergy between Multilingual Knowledge Completion, Entity Alignment and Relation Alignment. High confidence fact predictions may add valuable information for alignment tasks, and vice versa. Findings from experiments on multiple popular datasets indicate that AlignKGC achieves large improvements in KGC compared to a strong completion model that combines known facts in all languages. It also outperforms strong EA and RA baselines, underscoring the value of joint alignment and completion.

4.2 Proposed Methods

As mentioned before, AlignKGC is a multi-task system that learns to optimize for KGC, EA and RA simultaneously. This section discusses the loss components of the joint optimization objective.

4.2.1 KGC Loss

The KGC component in AlignKGC is an extension of the near state-of-the-art ComplEx [13] which defines a triples score as

$$f(s,r,o) = \Re\left(\langle s,r,o^{\star}\rangle\right) \tag{4.1}$$

where $c^{\star}$ is complex conjugate, $\langle\cdots\rangle$ is a 3-way elementwise inner product and $\Re(\cdot)$ is the real part of a complex number. Using f, ComplEx defines

$$\begin{aligned} \Pr(o \mid s,r) &= e^{f(s,r,o)} / \sum_{o'} e^{f(s,r,o')} \\ \Pr(s \mid o,r) &= e^{f(s,r,o)} / \sum_{s'} e^{f(s',r,o)} \end{aligned} \tag{4.2}$$

and the log-likelihood KGC loss as

$$L_{\text{KGC}} = \sum_{(s,r,o)\in \text{KG}} -\log \Pr(o \mid s,r) - \log \Pr(s \mid o,r) \tag{4.3}$$

4.2.2 RA Loss

To formulate the RA Loss term, the Hard SO-signature of a relation is defined as $\text{SO}(r) = \{(s,o) : (s,r,o) \in T\}$. The SO-overlap between two relations $r_l, r_{l'}$ is $|\text{SO}(r_l) \cap \text{SO}(r_{l'})|$. Jaccard similarity can then be used to compute a symmetric belief that two relations in different languages are equivalent:

$$b_J(r_l \Leftrightarrow r_{l'}) = \frac{|\text{SO}(r_l) \cap \text{SO}(r_{l'})|}{|\text{SO}(r_l) \cup \text{SO}(r_{l'})|} \tag{4.4}$$

If $b_J(r_l \Leftrightarrow r_{l'})$ exceeds a threshold θ (tuned hyperparameter), $(r_l, r_{l'})$ are added to the set $\mathscr{A}_J$ of 'silver' alignments.

To handle asymmetric relations, the belief score is modified to $b_A(r_l \Leftrightarrow r_{l'})$ defined as

$$b_A(r_l \Leftrightarrow r_{l'}) = \frac{|\text{SO}(r_l) \cap \text{SO}(r_{l'})|}{max\{|\text{SO}(r_l)|, |\text{SO}(r_{l'})|\}} \tag{4.5}$$

These ideas are then extended to re-define the SO-signature using entity embeddings, which can be trained via gradient descent. The Soft SO-signature is defined as $\text{SO}(r) = \{(\mathbf{s},\mathbf{o}) : (s,r,o) \in T\}$, where each element is the concatenatin of the subject and object embedding vectors. Then the embeddings pairs in these sets can be compared by extending cosine-similarity:

$$\text{sim}((\mathbf{s},\mathbf{o}),(\mathbf{s}',\mathbf{o}') = \sigma(\cos(\mathbf{s},\mathbf{s}')) \cdot \sigma(\cos(\mathbf{o},\mathbf{o}')) \tag{4.6}$$

Then the Soft-SO Overlap as the continuous extension of $|\text{SO}(r_l) \cap \text{SO}(r_{l'})|$, denoted by $\text{SoftOv}(r_l, r_{l'})$ is defined as the value of the maximal matching on the weighted bipartite graph induced by $A_{r_l,r_{l'}}$, where

$$A_{r_l,r_{l'}}[i,j] = \text{sim}(\text{SO}(r_l)[i], \text{SO}(r_{l'})[j]) \tag{4.7}$$

$b_{SA}(r_l \Leftrightarrow r_{l'})$ is defined by replacing terms in $b_A(r_l \Leftrightarrow r_{l'})$ with their 'soft' counterparts which leads to the soft asymmetric RA loss term for AlignKGC as shown below:

$$L_{\text{RA}-\text{SA}} = \sum_{(r_l,r_{l'})\in \mathscr{A}_{SA}} b_{SA}(r_l \Leftrightarrow r_{l'}) \|r_l - r_{l'}\|_1 \tag{4.8}$$

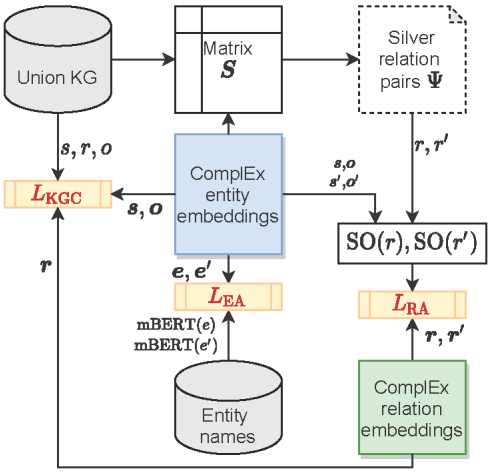

Figure 4.1: AlignKGC Architecture

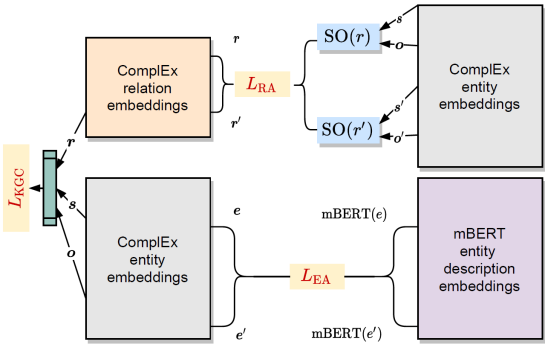

Figure 4.2: Loss components of joint objective

4.2.3 EA Loss

The loss component for entity alignment is

$$L_{\text{EA}} = \sum_{e_l, e_{l'}} \cos\left(\text{mbert}\left(e_l\right), \text{mbert}\left(e_{l'}\right)\right) \left\| e_l - e_{l'} \right\|_1 \tag{4.9}$$

where $\text{mbert}\left(e_l\right)$ is the textual embedding of entity e mapped in the KG space using a MLP. The complete AlignKGC loss term is then defined as

$$L_{\text{KGC}} + \alpha L_{\text{reg}} + \beta L_{\text{RA-SA}} + \gamma L_{\text{EA}} \tag{4.10}$$

where L_{reg} is an L_2 regularization on embeddings and $\alpha, \beta, \gamma \geq 0$ are tuned hyperparameters.

4.3 Alignment Inference

Entity Alignment: Entity Alignment is done by simply ranking the cosine similarity scores of the entity embedding of the entity to be aligned with the entity embeddings of all the entities in the target KG.

Relation Alignment: For relation alignment, the entity-aware relation matching protocol from RNM [21] is followed. The relation alignment inference mechanism for AlignKGC is mentioned below:

1. Both embeddings $\boldsymbol{r}_l, \boldsymbol{r}_{l'}$ and sets $\text{SO}(r_l), \text{SO}(r_{l'})$ can provide valuable signal toward the scoring of $(r_l, \equiv, r_{l'})$. Accordingly, let $S(r) = \{s : (s, \cdot) \in (r)\}$ and $O(r) = \{o : (\cdot, o) \in (r)\}$

be the subject and object entities involved with r, and let $\overrightarrow{S(r)}, \overrightarrow{O(r)}$ be the average of their entity embeddings. Following RNM, a relation r is represented as the concatenation $[\boldsymbol{r}, \overrightarrow{S(r)}, \overrightarrow{O(r)}]$. This is now treated as the embedding to be used for alignment of relations.

2. Cosine similarity matrix $\mathbf{S^r}$ is calculated using the previously determined relation representations for relation alignment. Thus s^r_{ij} is the cosine similarity score between representations of r_i and r_j.

3. Now the similarity scores are updated based on the intuition that if we have more alignments of head entities and tail entities at the same time in their associated triples, the more likely are two relations equivalent. For a relation r, $S_r = \{(h,t) \mid (h,r,t) \in T\}$ is defined as the set of its related entity pairs, where T denotes the set of triples in the given KG. Thus, given a candidate relation pair (r_i, r'_j), first the corresponding entity pair sets S_{r_i} and $S_{r'_j}$ are formed. Then, a matching set is defined as:

$$M^r_{ij} = \{(h_1,h_2),(t_1,t_2) \mid (h_1,t_1) \in S_{r_i}, (h_2,t_2) \in S_{r'_j}, s.t.\,(h_1,h_2) \text{ and } (t_1,t_2) \in \mathbb{L}_e\} \quad (4.11)$$

   where $\mathbb{L}_e$ is the set that includes the gold seed entity alignments and high confidence silver entity alignments based on a threshold hyperparameter for the cosine similarity scores.

4. The similarity between the relation pair (r_i, r'_j) is then updated as follows,

$$s^r_{ij} = s^r_{ij} + \lambda_r \cdot \frac{|M^r_{ij}|}{|S_{r_i}| + |S_{r'_j}|}, \quad (4.12)$$

   where λ_r is a tradeoff coefficient.

The performance of AlignKGC on various datasets for KGC, EA and RA tasks has been presented in detail in Chapter 5.

Chapter 5

Experiments

5.1 Setup

Some salient details about the experimental setup for training and testing AlignKGC are given below:

1. The experiments are conducted on the DBP-5L, the DBP15K and the OpenEA datasets. These datasets are described in Chapter 2. Note that only DBP-5L supports all three tasks (KGC, EA, and RA) whereas DBP15K and OpenEA support only EA and RA. But as AlignKGC aims to solve all three tasks, some fact triples from DBP15K and OpenEA are reserved for validation and test sets to create custom datasets that can support KGC along with EA and RA. To ensure fair comparison, other competing alignment baselines also do not have access to these reserved facts while training.

2. Based on grid search hyperparameter tuning, the following hyperparameters were chosen for AlignKGC: learning rate $= 0.8, \alpha = 0.02, \beta = 500$, and $\gamma = 1000$. 2000 negative instances are sampled for each positive triple when training ComplEx. The initial entity and relation embeddings are sampled from $\mathscr{N}(0, 0.05)$.

3. For competing baselines like RNM and RDGCN, several methods of initialization of GCN embeddings like translate+GloVE (default for RNM and RDGCN), mBERT, and translate+mBERT were compared. For both RNM and RDGCN, translate+GloVE was the best performing initialization.

4. For AlignKGC's text-based EA sub-model, translate+BERT was chosen as it performs better than mBERT.

Competing Baselines

1. **KGC:** KEnSb(RotateE) [4] and KG-BERT [19] are used as third-party baselines for KGC. Various versions of AlignKGC are also compared to demonstrate that each additional proposed component improves KGC performance.

2. **EA:** RNM and RDGCN are used as strong baselines for the entity alignment tasks. Both these methods make use of text signals.

5.2 KGC performance

Results of KGC experiments on DBP-5L, DBP15K and OpenEA are shown in Tables 5.1, 5.2, and 5.3 respectively. Both KGC baselines are outperformed by all variants of AlignKGC on DBP-5L. Performances of AlignKGC variants also show a similar pattern across all three datasets, and SoftAsym+Text consistently beats the other variants as expected.

	Greek (EL)			**English (EN)**			**Spanish (ES)**			**French (FR)**			**Japanese (JA)**		
	H@1	**H@10**	**MRR**	**H@1**	**H@10**	**MRR**	**H@1**	**H@10**	**MRR**	**H@1**	**H@10**	**MRR**	**H@1**	**H@10**	**MRR**
KGCmono	23.6	49.0	31.9	18.8	43.0	26.9	22.1	50.1	31.4	24.0	50.4	32.8	26.4	49.4	34.4
KG-BERT	17.3	40.1	27.3	12.9	31.9	21.0	21.9	54.1	34.0	23.5	55.9	35.4	26.9	59.8	38.7
KEnSb(RotatE)	27.5	56.5	-	14.4	39.6	-	25.2	62.6	-	22.3	60.6	-	32.9	64.8	-
KGCunion	25.1	56.2	35.0	18.3	43.7	26.4	22.6	52.7	32.3	26.0	54.7	35.8	29.3	55.9	38.6
Jaccard	25.1	57.8	35.7	18.5	45.0	27.2	23.0	52.3	32.7	27.6	58.0	37.6	32.9	59.8	41.9
Asymmetric	26.6	58.9	37.8	19.7	45.2	28.0	25.6	56.0	35.6	29.5	58.4	39.1	33.5	59.9	42.2
SoftAsym	32.7	61.0	42.5	20.8	45.9	29.1	26.9	55.9	36.4	30.7	59.3	40.3	34.7	61.7	43.9
Asym+Text	53.8	88.1	66.4	35.5	65.4	46.0	48.8	82.8	61.1	49.5	83.4	61.7	52.4	78.9	61.8
SoftAsym+Text	**57.6**	**88.4**	**69.0**	**37.2**	**66.1**	**47.4**	**53.0**	**84.4**	**64.5**	**52.9**	**84.9**	**64.5**	**53.2**	**80.9**	**62.9**
Unseen test set															
KGCunion	21.6	50.5	30.8	16.9	41.9	24.9	20.2	49.9	29.7	22.4	51.1	32.2	25.1	50.4	33.9
Jaccard	20.1	52.5	30.4	16.9	43.0	25.4	19.8	48.8	29.3	23.0	54.3	33.1	26.7	54.2	35.7
Asymmetric	20.3	52.8	31.3	17.5	43.0	25.7	21.9	52.5	31.8	24.6	54.3	34.4	27.1	53.8	35.8
SoftAsym	25.5	55.3	35.6	18.0	43.5	26.4	22.2	52.0	31.8	24.7	54.9	34.8	27.7	55.5	36.9
Asym+Text	48.7	86.2	62.2	33.2	63.9	43.9	45.8	81.3	58.5	45.4	81.6	58.4	47.0	75.6	57.0
SoftAsym+Text	**52.6**	**86.5**	**65.0**	**34.9**	**64.6**	**45.3**	**50.0**	**82.9**	**62.0**	**49.2**	**83.2**	**61.5**	**47.7**	**77.8**	**58.0**
Seen test set															
KGCunion	45.3	89.3	59.0	48.3	82.0	59.2	48.8	83.1	60.6	57.3	86.3	67.7	53.1	87.6	65.6
Jaccard	54.0	88.7	66.3	54.4	87.8	65.8	57.1	90.2	68.9	67.3	90.3	76.0	68.3	91.6	77.2
Asymmetric	62.7	94.0	75.5	68.5	94.8	78.4	65.4	94.6	76.8	72.4	94.2	79.8	70.2	94.7	78.8
SoftAsym	74.7	94.0	82.2	81.7	98.8	87.9	77.2	97.8	85.7	82.6	97.5	88.3	75.2	96.9	83.8
Asym+Text	83.3	99.3	90.3	85.9	99.4	91.7	81.6	99.3	89.3	**85.2**	98.6	**91.1**	82.9	97.8	89.4
SoftAsym+Text	**86.7**	**99.3**	**92.1**	**87.8**	**100.0**	**93.1**	**86.0**	**99.8**	**92.0**	*84.5*	**99.5**	*90.9*	**84.8**	**98.5**	**90.6**

Table 5.1: DBP5L, EA=30%, RA=0%, KGC performance. **Best,** *second-best* numbers.

	DBP15k-FR-EN						DBP15k-JA-EN						DBP15k-ZH-EN					
	French (FR)			English (EN)			Japanese (JA)			English (EN)			Chinese (ZH)			English (EN)		
	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR
KGCUnion	26.0	56.1	36.0	27.3	57.6	37.4	27.5	52.5	35.8	27.7	54.9	36.8	21.0	43.9	28.7	25.6	51.5	34.4
SoftAsym	27.4	57.0	37.2	28.7	58.9	38.6	28.7	53.1	36.9	28.7	54.6	37.2	22.5	45.4	30.2	26.8	51.9	35.3
SoftAsym+Text	**38.5**	**68.6**	**48.9**	**39.8**	**68.8**	**49.9**	**36.4**	**62.3**	**45.3**	**35.5**	**61.2**	**44.0**	**29.4**	**53.7**	**37.6**	**30.4**	**55.4**	**39.0**
Unseen test set																		
KGCUnion	25.7	55.8	35.7	27.0	57.2	37.1	27.1	52.1	35.4	27.4	54.6	36.5	20.1	42.8	27.7	25.0	50.7	33.7
SoftAsym	27.0	56.6	36.8	28.1	58.5	38.2	28.1	52.6	36.3	28.2	54.3	36.8	21.0	44.1	28.7	25.5	50.9	34.1
SoftAsym+Text	**38.0**	**68.3**	**48.5**	**39.3**	**68.4**	**49.4**	**35.8**	**61.8**	**44.7**	**35.0**	**60.8**	**43.5**	**27.8**	**52.4**	**36.1**	**29.0**	**54.4**	**37.7**
Seen test set																		
KGCUnion	57.6	90.9	69.1	51.7	90.7	65.1	61.7	88.3	70.5	56.8	83.2	66.2	50.5	82.0	62.2	49.3	86.3	61.4
SoftAsym	71.7	97.0	82.6	78.8	97.5	85.5	78.7	95.7	85.4	76.8	89.5	81.5	73.5	93.0	80.9	77.5	93.4	83.5
SoftAsym+Text	**88.9**	**100.0**	**93.2**	**89.8**	**100.0**	**94.8**	**88.3**	**97.9**	**92.4**	**83.2**	**96.8**	**88.9**	**83.0**	**97.0**	**89.2**	**87.2**	**98.2**	**91.9**

Table 5.2: DBP15K, revealed EA=30%, RA=0%, KGC performance.

	OpenEA-EN-DE-V1						OpenEA-EN-DE-V2						OpenEA-EN-FR-V1						OpenEA-EN-FR-V2					
	English (EN)			German (DE)			English (EN)			German (DE)			English (EN)			French (FR)			English (EN)			French (FR)		
	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR
	15.2	35.3	21.9	22.2	43.0	28.8	19.5	46.5	28.3	29.5	55.7	38.4	32.9	54.5	40.3	31.4	53.2	39.0	43.3	71.2	53.0	42.6	68.5	51.6
SoftAsym	15.8	35.2	21.9	22.5	43.7	29.5	20.0	46.8	28.8	28.4	54.1	37.0	33.9	55.0	40.9	31.5	53.6	39.2	43.2	70.4	52.7	42.0	69.1	51.3
SoftAsym+Text	**37.1**	**55.8**	**43.4**	**39.9**	**61.4**	**47.4**	**43.0**	**68.0**	**51.7**	**46.7**	**73.3**	**56.3**	**47.2**	**65.9**	**54.0**	**48.2**	**67.2**	**55.1**	**59.1**	**82.0**	**67.4**	**59.5**	**82.8**	**68.1**
Unseen test set																								
	14.3	34.2	20.9	21.3	42.1	27.9	18.5	45.5	27.3	28.9	55.1	37.8	31.8	53.4	39.2	30.5	52.1	37.9	42.3	70.4	52.1	41.5	67.6	50.6
SoftAsym	14.6	34.0	20.6	21.4	42.7	28.4	18.8	45.8	27.6	27.5	53.5	36.2	32.4	53.8	39.5	30.0	52.3	37.7	42.1	69.6	51.7	40.6	68.2	50.1
SoftAsym+Text	**36.0**	**54.9**	**42.4**	**38.9**	**60.8**	**46.5**	**41.9**	**67.3**	**50.7**	**46.0**	**72.9**	**55.7**	**45.9**	**64.9**	**52.8**	**46.9**	**66.3**	**53.9**	**58.0**	**81.5**	**66.5**	**58.2**	**82.2**	**67.1**
Seen test set																								
	62.8	96.5	74.2	74.7	95.4	83.2	69.2	94.8	78.7	71.2	95.0	79.4	72.4	93.7	80.9	62.5	91.7	74.0	79.2	96.6	86.6	74.7	97.0	83.4
SoftAsym	84.9	98.8	91.2	88.5	98.9	92.6	80.8	95.4	85.7	83.5	97.1	89.0	85.8	96.9	90.6	82.5	95.8	87.8	83.3	98.5	89.3	81.9	97.0	88.2
SoftAsym+Text	**96.5**	**100.0**	**97.8**	**94.3**	**100.0**	**97.1**	**95.4**	**99.4**	**97.4**	**94.2**	**100.0**	**97.0**	**96.1**	**100.0**	**97.7**	**91.7**	**99.2**	**94.8**	**95.5**	**100.0**	**97.7**	**97.0**	**99.6**	**98.3**

Table 5.3: OpenEA, revealed EA=30%, RA=0%, KGC performance.

5.3 EA performance

Results of EA experiments on DBP-5L, DBP15K and OpenEA are shown in Tables 5.4, 5.5, and 5.6 respectively. Translate+mBERT is a formidable baseline, beating both RNM and RDGCN on all three datasets. However, SoftAsym+Text clearly adds further value to Translate+mBERT, establishing a new state of the art among known EA methods.

LangPair	Translate +mBert		SoftAsym +Text		RNM		RDGCN	
	All	**!M**	**All**	**!M**	**All**	**!M**	**All**	**!M**
EL-EN	83.8	82.6	**90.3**	*89.6*	74.9	74.1	71.3	70.1
EL-ES	83.3	81.9	**91.1**	*90.5*	79.4	78.0	74.7	73.5
EL-FR	82.1	80.8	**90.9**	*90.3*	72.4	70.7	72.7	71.3
EL-JA	74.8	74.6	**89.3**	*89.3*	68.3	68.1	64.4	64.2
JA-EN	76.5	76.4	**88.5**	*88.5*	64.5	64.4	58.2	58.2
JA-ES	74.3	74.2	**88.9**	*88.8*	65.0	64.9	60.0	59.8
JA-FR	73.9	73.7	**88.9**	*88.9*	70.6	70.5	60.2	60.1
ES-FR	89.5	76.4	**96.1**	*91.7*	84.9	72.2	87.1	74.5
ES-EN	93.3	86.5	**96.8**	*94.1*	88.0	79.2	87.8	78.8
EN-FR	90.5	81.0	**95.3**	*91.2*	81.2	69.9	83.2	71.1
AVG	82.2	78.8	**91.6**	90.3	74.9	71.2	72.0	68.2

(!M = no exact match)

Table 5.4: DBP5L, EA H@1 test performance, 30% seed alignments.

LangPair	Text		SoftAsym +Text		RNM		RDGCN	
	All	!M	All	!M	All	!M	All	!M
fr-en	87.48	78.77	**90.31**	*84.86*	79.33	72.72	75.11	63.41
ja-en	64.29	63.40	**69.43**	*68.84*	62.29	62.05	50.44	49.88
zh-en	48.00	46.39	**55.10**	*53.82*	53.57	52.79	42.30	41.13
avg	66.59	62.85	**71.61**	*69.17*	65.06	62.52	55.95	51.47

(!M = no exact match)

Table 5.5: DBP15K, EA H@1 test performance, 30% seed alignments.

LangPair	Text		SoftAsym +Text		RNM		RDGCN	
	All	**!M**	**All**	**!M**	**All**	**!M**	**All**	**!M**
EN-DE-15K-V1	87.1	72.6	**89.7**	*78.7*	74.2	67.4	76.5	62.6
EN-DE-15K-V2	87.2	72.1	**91.6**	*82.2*	82.2	68.7	79.8	66.5
EN-FR-15K-V1	87.3	78.7	**90.5**	*84.9*	70.3	69.0	70.1	61.7
EN-FR-15K-V2	91.4	84.9	**93.3**	*88.2*	83.7	76.7	84.8	76.0
avg	88.2	77.1	91.3	83.5	77.6	70.5	77.8	66.7

(!M = no exact match)

Table 5.6: OpenEA, EA H@1 test performance, 30% seed alignments.

5.4 RA performance

Relations are split into rare (SO-set has under 500 SO pairs) and frequent ($\geq$500 SO pairs) relations. Naturally, RA predictions involving frequent relations are expected to be generally more accurate.

Results of RA experiments on DBP-5L, DBP15K and OpenEA are shown in Tables 5.7, 5.8, and 5.9 respectively. Asym+Text and SoftAsym+Text improve considerably over RNM, particularly for rare relations and H@1.

LangPair	Jaccard				Asymmetric				SoftAsym			
	Rare		Freq.		Rare		Freq.		Rare		Freq.	
	H@1	H@3	H@1	H@3	H@1	H@3	H@1	H@3	H@1	H@3	H@1	H@3
EL-EN	41.0	58.1	76.2	97.6	38.5	53.9	83.3	97.6	44.4	56.8	81.0	97.6
EL-ES	39.7	55.6	88.9	94.4	38.8	53.7	94.4	100.0	39.7	59.8	94.4	97.2
EL-FR	31.8	44.7	66.7	88.9	34.7	48.8	69.4	94.4	35.3	43.5	80.6	94.4
EL-JA	49.0	68.0	78.1	90.6	48.5	66.0	75.0	90.6	54.9	68.0	81.3	90.6
JA-EN	38.8	49.5	74.0	90.0	39.8	53.1	74.0	90.0	40.8	54.6	78.0	94.0
JA-ES	40.6	51.8	79.0	89.5	37.1	50.6	81.6	92.1	40.6	51.2	86.8	94.7
JA-FR	41.8	53.9	95.0	100.0	38.0	50.0	95.0	100.0	42.8	53.9	97.5	97.5
ES-FR	39.8	59.0	89.6	93.8	36.3	51.6	89.6	95.8	44.7	61.8	89.6	95.8
ES-EN	40.8	51.5	83.9	92.9	39.2	50.8	83.9	91.1	41.2	57.3	83.9	92.9
EN-FR	30.4	46.0	71.2	90.4	28.1	39.7	73.1	88.5	33.9	47.3	71.2	86.5
AVG	39.4	53.8	80.2	92.8	37.9	51.8	81.9	94.0	41.8	55.4	84.4	94.1
	RNM				**Asym+Text**				**SoftAsym+Text**			
EL-EN	58.1	72.2	90.5	97.6	71.4	85.5	92.9	100.0	74.4	85.9	95.2	100.0
EL-ES	69.2	79.4	94.4	97.2	79.9	86.0	94.4	100.0	83.2	88.8	94.4	100.0
EL-FR	53.5	62.9	75.0	94.4	62.4	73.5	83.3	97.2	62.4	74.1	83.3	94.4
EL-JA	74.8	83.5	81.3	90.6	82.5	88.8	84.4	93.8	83.0	89.8	84.4	93.8
JA-EN	54.6	64.8	86.0	96.0	61.7	75.0	82.0	94.0	63.8	76.5	82.0	94.0
JA-ES	46.5	58.8	84.2	92.1	56.5	67.1	84.2	94.7	57.7	74.7	84.2	94.7
JA-FR	70.2	75.0	92.5	92.5	78.9	88.0	92.5	100.0	79.3	87.0	92.5	97.5
ES-FR	75.5	84.5	89.6	95.8	82.6	87.9	91.7	95.8	84.2	87.0	91.7	95.8
ES-EN	64.2	75.8	89.3	94.7	77.7	85.4	85.7	98.2	80.0	86.9	85.7	98.2
EN-FR	46.9	63.4	75.0	88.5	58.9	69.6	76.9	92.3	60.3	71.0	75.0	90.4
AVG	61.3	72.0	85.8	93.9	*71.2*	*80.7*	*86.8*	**96.6**	**72.8**	**82.2**	**86.8**	*95.9*

Table 5.7: DBP5L, RA performance, 0% seed alignments.

	fr-en				ja-en				zh-en			
	Rare		Freq.		Rare		Freq.		Rare		Freq.	
Models	**H@1**	**H@3**	**H@1**	**H@3**	**H@1**	**H@3**	**H@1**	**H@3**	**H@1**	**H@3**	**H@1**	**H@3**
RNM	41.5	**50.9**	65.0	76.7	63.1	72.8	**79.2**	79.2	64.4	71.6	93.0	97.7
SoftAsym+Text	**45.0**	50.3	**71.7**	**80.0**	**68.1**	**78.9**	75.0	79.2	**66.3**	**74.1**	**96.5**	**100.0**

Table 5.8: DBP15K, RA performance, 0% seed alignments.

	EN-DE-15K-V1				EN-DE-15K-V2				EN-FR-15K-V1				EN-FR-15K-V2			
	Rare		Freq.		Rare		Freq.		Rare		Freq.		Rare		Freq.	
Models	**H@1**	**H@3**	**H@1**	**H@3**	**H@1**	**H@3**	**H@1**	**H@3**	**H@1**	**H@3**	**H@1**	**H@3**	**H@1**	**H@3**	**H@1**	**H@3**
RNM	69.9	78.9	92.3	92.3	68.2	80.7	90.9	**93.2**	71.2	81.0	94.1	94.1	72.2	81.9	94.7	94.7
SoftAsym+Text	**75.9**	**85.5**	92.3	92.3	**76.1**	**88.6**	90.9	90.9	**77.4**	**84.1**	**97.1**	**97.1**	**77.8**	**91.7**	94.7	**97.4**

Table 5.9: OpenEA, RA performance, 0% seed alignments.

5.5 mBERT vs GloVe representations for Entity Alignment

Experiments to compare mBERT with GloVE representations were conducted using the RNM model on the DBP15K dataset. Table 5.10 shows the results of these experiments. Note that for the changed initialisation, best hyperparameters were found using grid search.

	GloVe initialization (default)					mBERT initialization				
	EA			RA		EA			RA	
	Hits@1	**Hits@10**	**MRR**	**Hits@1**	**Hits@10**	**Hits@1**	**Hits@10**	**MRR**	**Hits@1**	**Hits@10**
FR-EN	93.79	97.94	0.9537	49.53	62.74	84.88	91.99	0.8764	46.23	58.49
EN-FR	93.73	98.33	0.9549	50	61.32	86.64	93.63	0.8931	45.28	55.66
JA-EN	86.41	94.51	0.8945	75.24	86.01	47.85	55.4	0.5074	58.22	74.48
EN-JA	86.54	94.97	0.8973	72.97	83.93	48.45	57.78	0.5190	52.55	65.78
ZH-EN	84.67	91.84	0.8739	81.01	87.75	54.05	61.17	0.5683	65.17	79.55
EN-ZH	83.41	92.26	0.8675	80.11	86.85	53.25	61.85	0.5658	61.35	72.81

Table 5.10: Comparison of GloVe and mBert initialisation for RNM

It is seen that GloVe outperforms mBERT for initialisation of GCN embeddings, but the most important observation is that the drop in performance is a lot more for language pairs with at-least one language with a non-latin root. This leads to the belief that mBERT inadequately represents these languages in its dictionary.

Chapter 6

Conclusions and Next Steps

The AlignKGC framework jointly learns 3 important tasks for Multilingual Knowledge Graphs, namely Knowledge Graph Completion, Entity Alignment and Relation Alignment. These three tasks have never been unified before. AlignKGC operates on the *KG* constructed by taking a union of all monolingual KGs, and extends KGC models to use novel EA and RA loss terms. In extensive experiments with three datasets, AlignKGC significantly improves KGC accuracy, as well as EA and RA accuracy, demonstrating the value of joint alignment and completion.

6.1 Next Steps

- The RA loss term for AlignKGC needs to be reformulated such that it can be included for every update instead of every few iterations.
- A new dataset which is more representative of real world data can be sampled from Wiki-Data, which should incude:
  - Ambiguity seen in real world data
  - Multilinguality as well as heterogeneity
  - One to many relationships
  - *Dangling* entities as characteristic of DBP2.0
  - Text descriptions of entities attribute triples

  This dataset would aim to establish a new benchmark for all KG related tasks discussed in this report.
- Implementation of a GNN-based version of AlignKGC which reformulates the KG as a tripartite graph, which has entities in the left layer and relations in the right layer with fact triples forming the middle layer.

References

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[2] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie 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.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[4] Xuelu Chen, Muhao Chen, Changjun Fan, Ankith Uppunda, Yizhou Sun, and Carlo Zaniolo. Multilingual knowledge graph completion via ensemble knowledge transfer, 2020.

[5] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015.

[6] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[7] Alvin E Roth. Deferred acceptance algorithms: History, theory, practice, and open questions. *international Journal of game Theory*, 36(3):537–569, 2008.

[8] Harkanwar Singh, Prachi Jain, Mausam, and Soumen Chakrabarti. Multilingual knowledge graph completion with joint relation and entity alignment, 2021.

[9] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706, 2007.

[10] Zequn Sun, Muhao Chen, and Wei Hu. Knowing the no-match: Entity alignment with dangling cases. *arXiv preprint arXiv:2106.02248*, 2021.

[11] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

[12] Xiaobin Tang, Jing Zhang, Bo Chen, Yang Yang, Hong Chen, and Cuiping Li. Bert-int: A bert-based interaction model for knowledge graph alignment. In *IJCAI*, pages 3174–3180, 2020.

[13] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR, 2016.

[14] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[15] Zhichun Wang, Jinjian Yang, and Xiaoju Ye. Knowledge graph alignment with entity-pair embedding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1672–1680, 2020.

[16] Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, Rui Yan, and Dongyan Zhao. Relation-aware entity alignment for heterogeneous knowledge graphs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, aug 2019.

[17] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

[18] Liang Yao, Chengsheng Mao, and Yuan Luo. Kg-bert: Bert for knowledge graph completion, 2019.

[19] Liang Yao, Chengsheng Mao, and Yuan Luo. Kg-bert: Bert for knowledge graph completion, 2019.

[20] Renbo Zhu, Meng Ma, and Ping Wang. Raga: Relation-aware graph attention networks for global entity alignment, 2021.

[21] Yao Zhu, Hongzhi Liu, Zhonghai Wu, and Yingpeng Du. Relation-aware neighborhood matching model for entity alignment, 2020.